DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TEMA: SEGUIMIENTO DE TRAYECTORIAS PARA UN CUADRICÓPTERO PARROT BASADO EN UN CONTROLADOR DIFUSO.

AUTOR: HEREDIA BARRAGÁN, DIDIER WALDEMAR

DIRECTOR: ING. PROAÑO ROSERO, VÍCTOR GONZALO

SANGOLQUÍ

2018

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación, **"SEGUIMIENTO DE TRAYECTORIAS PARA UN CUADRICÓPERO PARROT BASADO EN UN CONTROLADOR DIFUSO"**, fue realizado por el señor **HEREDIA BARRAGÁN, DIDIER WALDEMAR,** el mismo que ha sido revisado en su totalidad, analizado por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 09 de Agosto de 2018

Ing. Víctor Gonzalo Proaño Rosero.

C.C.: 1706457924

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

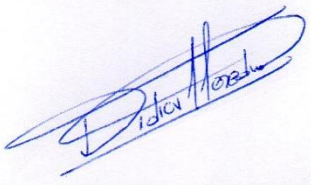
CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORÍA DE RESPONSABILIDAD

Yo, **HEREDIA BARRAGÁN, DIDIER WALDEMAR**, declaro que el contenido, ideas y criterios del trabajo de titulación: **SEGUIMIENTO DE TRAYECTORIAS PARA UN CUADRICÓPTERO PARROT BASADO EN UN CONTROLADOR DIFUSO,** es de mi autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando en las citas bibliográficas.

Consecuentemente el contenido de la investigación mencionada es veraz.

Sangolquí, 09 de Agosto de 2018



Didier Waldemar Heredia Barragán.

C.C.: 1714552658

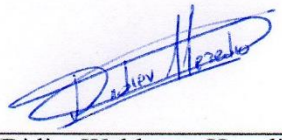
DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN

Yo, **HEREDIA BARRAGÁN, DIDIER WALDEMAR**, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **SEGUIMIENTO DE TRAYECTORIAS PARA UN CUADRICÓPTERO PARROT BASADO EN UN CONTORLADOR DIFUSO** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Sangolquí, 09 de Agosto de 2018



Didier Waldemar Heredia Barragán.

C.C.: 1714552658

DEDICATORIA

Este trabajo quiero dedicarlo a Dios por darme la fuerza y fortaleza para poder cumplir cada uno de mis sueños y metas propuestas.

A mi madre Catalina, por brindarme ese amor incondicional, ser mi mentora y protectora que estuvo para celebrar mis triunfos y derrotas, dándome palabras de aliento para seguir adelante y no darme nunca por vencido, por ser ese ser tan maravilloso al que siempre he admirado.

A mi padre Edwin, por ser mi mentor y mejor amigo, por estar a mi lado siempre apoyándome, siendo ese ejemplo de rectitud, honestidad y buenos valores que me han convertido en la persona que soy hoy en día.

A mis hermanos Estefany y Erick, por estar a mi lado apoyándome y dándome palabras de aliento para seguir adelante, por ser su amor y cariño incondicional, y por ser los mejores hermanos a quienes quiero y admiro mucho.

A mi novia Samantha, por ser mi compañera y amiga, por permitirme formar parte de su vida y permitirme brindarle todo mi amor, y ser mi apoyo incondicional en las buenas y malas.

A mi abuelita Aida, por haber sido una parte importante en mi vida, por ser ese ser maravilloso al que tuve el honor de llamar madre, por todo el cariño y amor que me supo dar y espero que este muy orgullosa del hombre en el que me he convertido.

AGRADECIMIENTO

Agradezco a Dios por bendecirme con una familia tan maravillosa, la cual siempre ha estado a mi lado y con la que tengo el honor de compartir este logro profesional.

A mi familia por estar siempre a mi lado, ser el pilar fundamental de mi vida, por compartir esos bellos momentos juntos y por inculcarme todos los valores que me convirtieron en el hombre que soy el día de hoy y por enseñarme el amor y gracia de Dios.

A mi novia Samantha, por ser mi compañera y apoyo durante todo este tiempo, por estar a mi lado en los momentos más difíciles de la carrera, y darme su amor incondicional siempre. Gracias por todo mi amor.

A mis compañeros de la universidad, Fabricio, Richard, Juan Carlos, Cristian, Bryan, por todas las buenas experiencias, por esa amistad y apoyo incondicional que me supieron brindar a lo largo de la carrera.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

Los vehículos aéreos no tripulados conocidos como UAV por sus siglas en inglés, o drones en la actualidad tiene muchas áreas de aplicación que pueden ir desde la seguridad y defensa, hasta el ocio o recreacionales. La mayoría de drones comerciales utilizados para recreación o fines lúdicos tiene la característica de ser manejados por un operador, es por este motivo que en el presente trabajo de investigación se desarrolló un sistema de seguimiento de trayectoria basado en controladores de lógica difusa. El proyecto se dividió en: comunicación entre el dron y el ordenador utilizando el meta sistema operativo Robot Operating System o ROS y Matlab; extracción de características de las imágenes captadas por el dron, en esta sección se recurrió al uso de la cámara integrada que posee el dron y al uso de librerías de visión por computadora; desarrollo de los controladores basados en lógica difusa para las variables de control de Roll y Yaw para gobernar los movimientos que realice el dron; diseño de la interfaz HMI la cual permite realizar el control y monitoreo del proyecto. Se realizaron pruebas en interiores y exteriores con el fin de comprobar la eficiencia del algoritmo en estos dos ambientes, obteniendo por resultado una eficiencia para completar la trayectoria en interiores del 86.66% y una eficiencia para completar la trayectoria en exteriores del 73.33%.

PALABRAS CLAVE:

- **SEGUIMIENTO DE TRAYECTORIA**
- **CONTROLADORES DIFUSOS**
- **HMI**

ABSTRACT

Unmanned aerial vehicles UAVs or drones currently have many fields of application that can range from security and defense, to leisure or recreational activities. The most of commercial drones are being used for recreational or learning purposes and it has the characteristic of being handled by an operator, so in this investigation research works on trajectory tracking system who is based on fuzzy logic controllers. The present project was divided first, communication between the drone and the computer using programs systems such as Robot Opetaring System (ROS) and Matlab; Second, extraction of characteristics of the images captured by the drone, and in this section we must to use the integrated camera that owns the drone also computer vision libraries; third, development of controllers based on fuzzy logic, for the control variables of Roll and Yaw angles, to control the movements made by the drone; Fourth, design of the HMI interface which allows the control and monitoring of the project. Indoor and outdoor tests were carried out in order to verify the efficiency of the algorithm in these two environments, as a result we obtain an efficiency to complete the interior trajectory of 86.66%, and we obtain an efficiency to complete the exterior trajectory of 73.33%.

KEYWORDS:

- **PATH FOLLOWER**
- **FUZZY LOGIC CONTROLER**
- **HMI**

CAPÍTULO I

1. Introducción

1.1 Antecedentes

Los UAVs o Vehículos Aéreos No Tripulados y sus aplicaciones en distintas áreas como la industria manufacturera, la medicina, la exploración, aplicaciones multimedia e incluso aplicaciones militares han adquirido gran importancia en los últimos años (Kore, 1998), (Kendoul, 2012). El desarrollo tecnológico de Vehículos Aéreos no Tripulados UAV ha ganado interés en áreas como cartografía, vigilancia, búsqueda y operaciones de seguimiento (Mojica, 2015), (Tisdale, 2009). Las primeras aplicaciones de los UAVs fueron en el campo militar. Sin embargo en la actualidad se han realizado investigaciones para su uso en otras áreas como en la industrias de fabricación de juguetes, industria del entretenimiento, entre otras (Kore, 1998), (Barrientos, 2012).

Dentro de los UAVs se puede hablar sobre los sistemas para el control de vuelo, navegación y los sistemas de guiado, (Kendoul, 2012) .En su investigación, los sistemas de control de vuelo se clasifican en tres diferentes: a) Sistemas de control de vuelo basados en aprendizaje, b) Sistemas lineales de control, y c) Modelos basados en controladores no lineales. Además del control de vuelo también se requiere de un sistema de navegación para la detección, estimación del estado, percepción del medio ambiente y conciencia de la situación para conocer el ambiente. Adicionalmente al sistema de control de vuelo y al sistema de navegación se dispone de sistemas de guiado compuesto por: a) generación de trayectorias, b) planificación de trayectorias y evasión de obstáculos, c) planificación de misiones y d) razonamiento y toma de decisiones de alto nivel. Según el nivel de autonomía del UAV, cada uno de estos componentes puede ser realizado manualmente por el operador o de forma autónoma por el vehículo.

Una de las innovaciones a nivel comercial de estos sistemas es la implementada por la empresa de compras por Internet Amazon, la cual realiza entregas de paquetes con el uso de UAVs (Gutierrez, 2017). En la actualidad este servicio se encuentra disponible en el servicio de *Amazon Prime Air*, el cual solo entrega paquetes de hasta 5 libras de peso máximo, (Amazon, 1996-2017). Otra empresa que se ha sumado a esta tendencia es UPS, la cual anunció que ha probado satisfactoriamente la entrega de paquetes mediante un UAV lanzado desde la parte superior de una furgoneta, (Juste, 2017).

Dentro de la línea de investigación de los sistemas de control de vuelo, y en el subtema de controladores basados en aprendizaje, se han realizado diversas investigaciones al implementar diferentes sistemas de control. Uno de estos sistemas de control es un PID (Proporcional Integral y Derivativo) ajustable como expone (Sangyam, 2010). En su investigación se comparó un controlador PID convencional y un controlador PID ajustable basado en lógica difusa, el PID autoajustable basado en lógica difusa es adecuado para sistemas que sufren de parámetros inciertos, como la variación en la carga útil, ya que el algoritmo puede ajustar la ganancia de los controladores PID. Otra comparación entre controladores la realizaron Bora, (Bora, 2012).En su investigación exponen la comparativa entre un controlador PD (Proporcional Derivativo) y un controlador Fuzzy PD. Ambos controladores fueron capaces de controlar al sistema, pero el controlador fuzzy PD funcionó mejor con respecto al rechazo a las perturbaciones además presentó mayor facilidad en su construcción.

La lógica difusa se puede utilizar para realizar un seguimiento de ruta como expone. (Dong, 2005) . En su trabajo se implementó esta técnica para el seguimiento de ruta de un UAV en conjunto con la evasión de obstáculos, los cuales se consideraron como móviles, inmóviles y aleatorios a lo largo de la ruta de trabajo. Otra aplicación es el control de la altitud de un UAV

como expone , (Younes, 2008) . Este autor presenta la implementación de un controlador BLF o *Backstepping Fuzzy Logic controller* por sus siglas en inglés, y un controlador BLMS o *Backstepping Least Mean Square controller* por sus siglas en inglés, con el fin de dar un nuevo enfoque hacia el control de la altitud del UAV en torno a un punto de referencia, siendo el controlador BLMS el que tuvo una respuesta de tiempo más lenta.

De acuerdo a los antecedentes, y retos que todavía se tiene dentro de la temática de los sistemas de control de vuelo se propone en esta investigación el diseño e implementación de un sistema de control de trayectoria utilizando controladores basados en lógica difusa.

1.2 Justificación e Importancia.

Las primeras invenciones relacionadas con los vehículos aéreos no tripulados aparecieron en 1980. A partir de 1997 y hasta el presente, la tecnología ha estado en constante crecimiento y su evolución caracterizada por un alto número de competidores y desarrollo de invenciones donde se encuentran países como: Estados Unidos, Alemania, Francia y Reino Unido. Estados Unidos es el país líder en actividad inventiva con 633 invenciones (Mojica, 2015).

En el Ecuador también se han realizado proyectos con los UAV´s enfocados en dar soporte audiovisual, soporte para aplicaciones militares, mejoramiento de la agricultura, evasión de obstáculos o para el control automático de robots (Sandoval, 2016) (Terán, 2016). Por ejemplo, para mejorar las plantaciones de cualquier cultivo se han utilizado drones con cámaras multiespectrales instaladas en el ala fija para la detección de estrés vegetal, distribución hídrica, plagas y coordenadas de áreas de fumigación a través de fotografías obtenidas durante el sobrevuelo de las plantaciones (Terán, 2016). Otra aplicación que está en crecimiento es en la fotografía dentro de la cual se utiliza para dar apoyo audiovisual (Vargas, 2014).

Dentro del ámbito militar en el encuentro científico “Avances y Aplicaciones Tecnológicas de Drones y Sistemas Aéreos no Tripulados” el Mayor Lenin Jara, Director CIDFAE *(Centro de Investigación y Desarrollo de la FAE)* en representación de la Dirección de Desarrollo Aeroespacial, resaltó que las aplicaciones tecnológicas dentro del campo de los UAVS es de trascendencia nacional y aporta a los objetivos permanentes y estratégicos a las Fuerzas Armadas, Defensa nacional y Segundad integral, para disminuir la dependencia tecnológica extranjera y orientar los esfuerzos del Centro de Investigación a la defensa del territorio nacional (EPN, 2017).

La aplicación de los UAV’s en el Ecuador se encuentra en desarrollo especialmente en áreas como la seguridad, cartografía y agricultura. Por tal motivo, se propone desarrollar un sistema para control de la trayectoria de un UAV, empleando un controlador basado en lógica difusa, con la finalidad de que este trabajo sirva como soporte en estas áreas de aplicación sin la interferencia de la experticia del operador o de un controlador especializado.

1.3 Alcance del Proyecto.

En el presente trabajo de investigación se desarrollará un sistema para el seguimiento de trayectorias con un controlador basado en lógica difusa, el cual se implementará un UAV tipo cuadricóptero de Parrot.

En la etapa inicial del trabajo de investigación se realizará la instalación de ROS o *Robot Operating System* por sus siglas en inglés. Después se realizará la instalación de Matlab en Ubuntu, para realizar la comunicación entre ROS y Matlab con la finalidad de aprovechar todas las herramientas disponibles de este software.

En la siguiente etapa del proyecto, se procederá a realizar la identificación de las variables de interés dentro del UAV que estarán ligadas al movimiento, como los ángulos Roll, Pitch y Yaw, y

como siguiente paso se realizará el controlador basado en lógica difusa con el fin de realizar el control de una trayectoria.

A continuación, se realizará la interfaz HMI o *Human Machine Interface* por sus siglas en inglés, esta se desarrollará con la aplicación de Matlab *app designer y* constará de un menú para: el comando de despegue y aterrizaje del UAV; inicio y paro de la aplicación y visualización de la posición actual del UAV dentro del espacio de trabajo.

La última etapa del trabajo de investigación se enfocará en la implementación de los pasos anteriores dentro del UAV y observar resultados obtenidos dentro del proyecto, con el fin de realizar una retroalimentación para trabajos futuros.

1.4 Objetivos

1.4.1 Objetivo General

- Desarrollar un sistema para el seguimiento de una trayectoria basado en lógica difusa para el UAV tipo cuadricóptero de Parrot.

1.4.2 Objetivos Específicos

- Implementar un sistema de comunicación entre el vehículo aéreo no tripulado y el ordenador, basándose en la plataforma ROS.
- Identificar las variables de interés dentro del UAV para realizar el controlador difuso.
- Desarrollar el controlador basado en lógica difusa para realizar el seguimiento de la trayectoria.
- Diseñar la interfaz HMI que permita realizar la visualización y monitoreo del proyecto

CAPÍTULO II

2. Estado del Arte

2.1 Origen de los Vehículos Aéreos No Tripulados

El origen de los vehículos aéreos no tripulados o UAV por sus siglas en inglés , no es relativamente nuevo, desde hace muchos años atrás se realizaron los primeros intentos para llevar a cabo vehículos no tripulados para portar una carga útil, como es el caso del ejercito austriaco el cual envió varios globos cargados con explosivos hacia la ciudad de Venecia (Rodriguez, 2016).

En años posteriores se continuaron realizando invenciones como es el torpedo Kettering construido por General Motors, el cual era un biplano que podía desplegar sus alas y caer en un lugar determinado (Rodriguez, 2016). Después el desarrollo de los UAV se enfocó en la rama de la milicia, desarrollando aeronaves que podían ser pilotadas remotamente, servían para misiones de reconocimiento y exploración, constaban con diversos sensores y armamento.

En el año del 2010 es cuando se universaliza el uso de los UAV de pequeño tamaño, en el ámbito civil, tanto para objetivos comerciales como para el ocio, es así que diversos países empiezan a crear leyes y regulaciones para restringir el uso de este tipo de vehículos, para su utilización en espacios aéreos abiertos (Rodriguez, 2016).

2.2 Clasificación de los Vehículos Aéreos no Tripulados

Los vehículos aéreos no tripulados, en la actualidad tienen muchas áreas de aplicación como lo es la filmación de video, entrega de paquetes, seguridad y defensa, competiciones deportivas o la milicia, por todo ello, la industria de los drones está en pleno apogeo, con tasas de crecimiento exponenciales año tras año (Fraga, 2017).

Dentro de la gama que comprenden los vehículos aéreos no tripulados tenemos cinco categorías según el tamaño y la carga útil (Kendoul, 2012):

- Categoría I: escala completa; Las características principales son la robustez de la estructura física y la carga útil que pueden transportar.
- Categoría II: escala media; Tienen una carga útil superior a 10 kg y un peso total superior a 30 kg.
- Categoría III: pequeña escala; Estos UAV tienen una carga útil de 2 a 10 kg, con un peso total inferior a 30 kg.
- Categoría IV: mini; Tienen una carga útil de 2 kg, son operados eléctricamente, de bajo costo, tienen fácil mantenimiento y operación segura.
- Categoría V: Los vehículos Micro Air (MAV) tienen una carga útil inferior a 100 gramos y se utilizan en la navegación y detección.

Otra clasificación que se tiene de los UAV es según su construcción o tipo de fuselaje, estos se pueden clasificar en (Blyenburgh, 2006):

Ala Fija: son aquellos UAV en los cuales las alas se encuentran unidas al fuselaje y a su vez con todos los elementos de vehículo, dependiendo de la ubicación de las alas se pueden dividir en:

- Ala alta
- Ala Media
- Ala Baja
- Ala Volante

Ala Rotatoria: es el UAV dentro del cual las alas o palas giran alrededor de un eje, dependiendo del número de rotores o de su configuración se pueden dividir en:

- Rotor principal y rotor en cola

- Rotor único o Singlecopter
- Rotor en configuración Coaxial
- Rotores en configuración tándem
- Multirrotores

También existe una clasificación más de los UAV, a la que se conoce como UAV híbridos, estos son capaces de realizar despegues y aterrizajes de manera vertical como aeronaves de ala fija y son capaces de realizar vuelos a alta velocidad como un ala fija tradicional (Santana, 2017), en la Figura 1 se muestra un UAV ala fija.

Figura 1. UAV de Ala Fija
Fuente: (Santana, 2017)

En los UAV de ala rotatoria dentro de la categoría de multi rotores tenemos varias configuraciones que van desde el número de rotores que pueden ser tres, cuatro, seis u ocho rotores, y también la disposición que estos tengan, la cual puede ser en forma de cruz, x, H o Y. En la Figura *2* se muestran las diferentes configuraciones que se pueden tener de los UAV multirrotores.

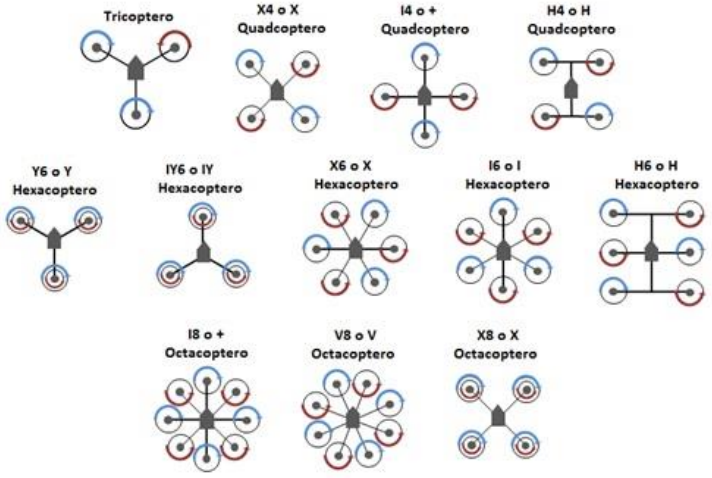


Figura 2. Configuraciones de UAV Multirrotores
Fuente: (Santana, 2017)

2.3 Sistemas de Control de Vuelo para Vehículos Aéreos no Tripulados

Los sistemas de control son una parte fundamental para los UAV, ya que estos son los encargados de gobernar los actuadores y a su vez el comportamiento que tendrá. Diversas técnicas de control son aplicadas con el fin de obtener un comportamiento óptimo del UAV. Estos sistemas son capaces de controlar la altura, velocidad y posición.

Los controladores de vuelo se pueden clasificar en tres categorías principales (Kendoul, 2012):

- Métodos de control basados en el aprendizaje.
- Sistemas Lineales de Control de vuelo.
- Controladores no lineales basados en modelos.

En los métodos de control basados en Aprendizaje podemos encontrar que las técnicas más características son:

- Lógica Difusa.
- Redes Neuronales.
- Aprendizaje basado en humanos.

En los sistemas de control basados en sistemas lineales podemos encontrar las siguientes técnicas de control como lo son:

- PID o Proporcional Integrador Derivador.
- LQR o Linear quadratic regulator por sus siglas en ingles.
- H∞.
- Programación de Ganancias o Gain Scheduling por sus siglas en ingles.

Y en los sistemas de control no lineales basados en modelos podemos encontrar las siguientes técnicas de control:

- Linearización en retroalimentación.
- Modelos de control predictivo.
- Saturaciones anidadas.
- Control Adaptativo.
- Backstepping.

En el campo de los métodos de control basados en aprendizaje, los controladores basados en lógica difusa tienen aspectos favorables en relación a los otros controladores, debido a que estos poseen una gran facilidad al momento de implementarlos, ya que permiten realizar las diversas acciones de control utilizando reglas (Santos, 2014).

Una de las ventajas que el controlador difuso presenta es que una vez ajustado correctamente, proporciona un rendimiento equilibrado para lograr la estabilización con una respuesta aceptable del sistema (Younes, 2008). La aplicación de estrategias inteligentes como la lógica difusa en el diseño de sistemas de control permite flexibilidad y eficiencia. La aplicación de conocimiento de un experto sobre el comportamiento a veces es la única forma de tratar con sistemas complejos (Santos, 2014).

Al realizar una comparación entre los sistemas de control más comunes como un PID, y un Fuzzy PID, con el fin de controlar la altura del dron, se expuso a ambos sistemas a una señal que representa la turbulencia, dando por resultados que el controlador Fuzzy PID posee una mejor respuesta ante la señal de turbulencia que el controlador PID clásico (Abbasi E. , 2014). Otra comparativa es aquella en la cual compara un controlador PID clásico y un controlador Fuzzy PID con el fin de estabilizar un dron modelado con la ecuación de Euler Newton, los resultados mostraron que el controlador Fuzzy PID presentó una notable mejora frente al PID con respecto a la eliminación del sobre impulso y en el tiempo de establecimiento (Abbasi E. , 2015).

2.4 Vehículos Aéreos no Tripulados comerciales

Los UAV (Unmanned Aerial Vehicle) sufren hoy en día una revolución con la creación de aeronaves cada vez más pequeñas y baratas. Tanto es así que en los últimos años han aparecido nuevas tecnologías que permiten el desarrollo y la comercialización de aeronaves de bajo coste para uso personal y profesional (Lavid, 2017).

A nivel mundial existen empresas que se encargan de la fabricación y venta de UAV o Drones, es así que muchas de estas han visto en los UAV una oportunidad de negocios, con un gran número áreas de aplicación. Podemos encontrar diferentes variedades de UAV desde los más profesionales, con sofisticados sistemas de control y cámaras profesionales, los UAV que son utilizados para carreras, que son personalizados por cada usuario y al final podemos encontrar UAV para fines lúdicos, o para desarrollo de aplicaciones.

La empresa líder dentro del mercado de los UAV es Dji de origen chino, seguida muy cerca por la empresa francesa Parrot, como se puede apreciar en la Figura 3 donde se muestran las 20 mejores empresas a nivel mundial referidas al año 2016.

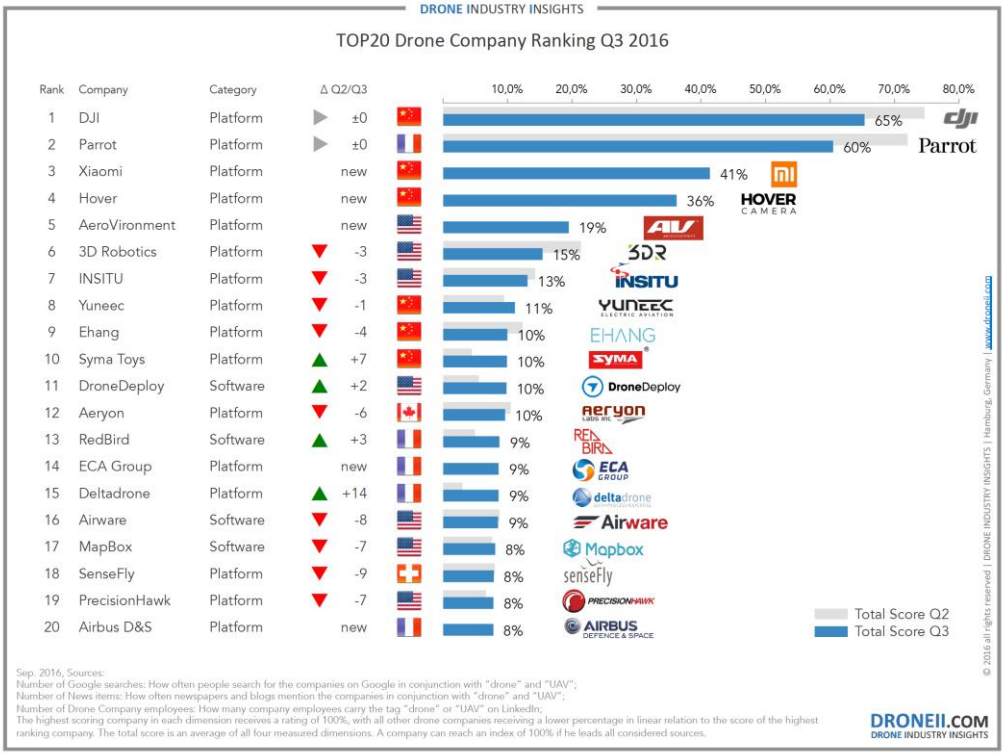


Figura 3. Empresas Comericales de UAV's a nivel mundial
Fuente: (Wackwitz, 2016)

El UAV más representativo de la empresa Dji es el "Phantom" que está en su cuarta versión, en la Figura 4 se muestra un fotografía. El UAV tiene una configuración mutirrotor de 4 hélices en x, una cámara profesional con estabilizador de imagen, y su propulsión es eléctrica, llegando a tener una autonomía de vuelo alrededor de 35 min.

Figura 4. Dji Phanton 4 pro
Fuente: (DJI, 2018)

La empresa francesa Parrot, tiene una amplia variedad de UAV's que van desde mini drones, drones, equipos de audio y para automóviles. Uno de los drones más representativos de esta

empresa es el Bebop, que en la actualidad se encuentra en su tercera versión. La Figura 5 muestra la fotografía del dron, Este dron cuenta con una configuración de ala rotatoria multirotor, que consta de 4 helices en una configuración en X.

Figura 5. Parrot Bebop
Fuente: (Parrot, 2018)

Una de las principales características para la versión Bebop 1 y Bebop 2 es poseer un SDK o *Software Development Kit* por sus siglas en inglés, el cual posee las herramientas necesarias para establecer comunicación con el dron, y hacer uso de los sensores que posee. Gracias a este SDK se puede desarrollar diversas aplicaciones o programas con el fin de implementar de diversos sistemas de control o aplicaciones puntuales.

A continuación, en la Tabla 1 se detallan ciertas características que posee el dron Bebop 1 de Parrot, frente a su predecesor el AR.Drone 2.0 de la misma empresa. Se realiza esta comparativa con el fin de apreciar cuál de estos drones posee las mejores características de desempeño, así como de autonomía de vuelo, cabe recalcar que ambos drones poseen un SDK, por lo cual los hace plataformas ideales para el desarrollo de aplicaciones.

Tabla 1

Comparativa entre AR.Drone 2.0 y Bebop 1.

	AR.Drone 2.0	Bebop 1
Cámara	HD 720p	Fish Eye 186°, 1080p
Procesador	1GHz ARM Cortex A8	2GHz ARM Cortex A9
Memoria	1Gb	8Gb
Batería	11.1V 1000mAh	11.1V 1100mAh
Tiempo de Vuelo	15 min	22 min
Acelerómetro	Si	Si
Magnetómetro	Si	Si
Giroscopio	Si	Si
GPS	No	Si

Fuente: (Parrot, 2018)

CAPÍTULO III

3. Plataforma ROS y Comunicación con Dron Bebop de Parrot

3.1 Introducción

Dentro de este capítulo se habla sobre el Robot Operating System o ROS por sus siglas en inglés, las características que este posee y diversas herramientas que lo componen. De igual manera se habla sobre el SDK necesario para realizar la comunicación entre ROS y el Cuadricóptero Bebop.

3.2 Robot Operating System

Robot Operating System, es un sistema operativo enfocado a la robótica, permite desarrollar software para diversos robots comerciales con mucha flexibilidad. Se basa en un conjunto muy amplio de herramientas, librerías y convenciones las cuales son de gran ayuda al momento de crear un programa complejo y robusto para poder aplicarlo en diferentes plataformas (ROS, 2018).

ROS se construyó desde cero para fomentar el desarrollo de software de robótica colaborativa, es decir que el conocimiento sea libre y gratuito. Esto se logra gracias a diversos investigadores alrededor del mundo que trabajan con la plataforma ROS, los cuales aportan su software y así puede ser utilizado por otra persona en cualquier parte del mundo.

ROS es un meta-sistema operativo de código abierto. Proporciona los servicios que esperaría de un sistema operativo, incluida la abstracción de hardware, el control de dispositivos de bajo nivel, la implementación de la funcionalidad de uso común, el paso de mensajes entre procesos y la gestión de paquetes. También proporciona herramientas y bibliotecas para obtener, construir, escribir y ejecutar código en múltiples computadoras (ROS, 2018).

3.2.1 Herramientas de ROS

ROS tiene mucho potencial debido a las herramientas y librerías que lo componen. Fue concebido para ser distribuido y modular, de modo que sea posible elegir las herramientas y

librerías a utilizar en función de la necesidad que la aplicación lo requiera, de este modo se ahorran recursos del ordenador.

A continuación, se presentan los conceptos básicos que forman parte de los procesos de ROS (ROS, 2018).

- **Nodos:** Son los encargados de realizar algún tipo de cálculo computacional en el sistema. ROS está estructurado modularmente, de manera que puede comprender diversos nodos que controlen diversas partes de un robot.
- **Master:** Es el nodo principal, este proporciona el registro de nombres y de consulta para la computación gráfica. Sin este, los nodos son incapaces de trabajar, ya que no encontrarían los mensajes necesarios o los servicios que la aplicación requiera.
- **Parameter Server:** Permite el almacenamiento de datos en una localización central.
- **Messages:** Sirve para la comunicación entre nodos, poseen una estructura de datos que puede ser numérica y de datos anidados.
- **Topics:** Son canales de información entre los nodos, esta comunicación se realiza por medio de mensajes. Un nodo puede publicar o suscribirse a un tópico, para obtener información, y a su vez un mismo nodo puede estar suscrito a varios nodos.
- **Services:** Proporcionan comunicación mediante un método de pregunta respuesta, cuando un nodo quiere acceder a un servicio, tiene que llamarle y esperar una respuesta. Un nodo solo puede llamar a un servicio.
- **Bags:** son un formato para guardar y reproducir datos de mensajes de ROS. Son un mecanismo importante para almacenar datos, como datos de sensores.

En la Figura 6 se puede apreciar cómo se realiza la comunicación entre los nodos del driver bebop autonomy y el nodo de Matlab que utilizaremos en este proyecto. El nodo Master habilita

procesos de identificación y comunicación mutuas entre los nodos, simplemente se tiene que ingresar el nombre del topic que desea publicar o suscribirse, y el maestro se encargará de realizar la comunicación.

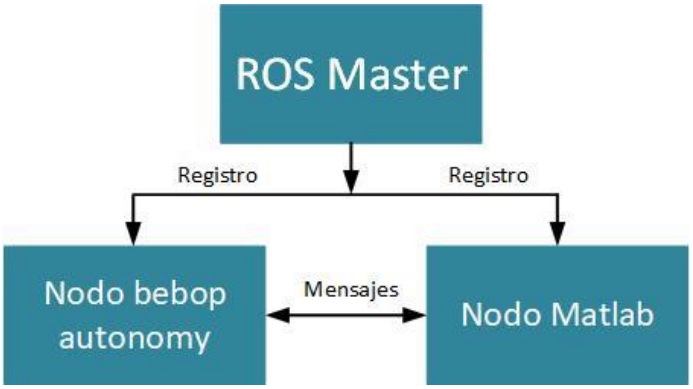


Figura 6. Comunicación entre nodos de ROS

Todos los nodos de ROS se comunican directamente unos con otros, el maestro proporciona información de búsqueda, al igual que un servidor DNS. Todos los nodos que se suscriben a un topic solicitaran conexión con los nodos que realizan publicación de ese topic, estableciendo así una conexión como se puede apreciar en la Figura 7, la comunicación entre nodos se realiza por medio del protocolo de comunicación utilizado en ROS el cual es el TCPROS. Este es una capa de transporte para los mensajes y servicios de ROS, utiliza los conectores TCP/IP estándar para transportar los datos de mensajes (ROS, 2018).

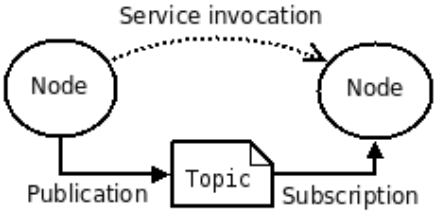


Figura 7. Flujo de Información entre nodos de ROS
Fuente: (ROS, 2018)

3.2.2 Ejecución y comandos de ROS

ROS proporciona diversas herramientas las cuales permiten simular y visualizar entornos y robots, también permite ejecutar varios nodos en un mismo tiempo y apreciar el flujo de información entre estos. A continuación, se presentan varias herramientas más utilizadas dentro de ROS (ROS, 2018).

- **roscd:** Permite acceder a un paquete, localidad de pila o a una locación en común.
- **roscore:** Ejecuta el núcleo de ROS, es muy importante ya que con este se inicializa la comunicación entre nodos, servicios, parámetros, mediante el nodo master.
- **rosdep:** permite instalar dependencias del sistema.
- **roscreate-pkg:** Con esta herramienta el usuario puede crear los paquetes que necesite, además de especificar si posee alguna dependencia de otro paquete.
- **rosrun:** Permite ejecutar un archivo ejecutable en un paquete arbitrario.
- **roslaunch:** Puede ejecutar archivos con formato. launch, en estos se puede configurar a varios nodos para ejecutarlos al mismo tiempo.
- **rosmsg:** Permite obtener la información de un tipo de mensaje.
- **rosnode:** Muestra la lista de nodos en ejecución, y permite realizar ping para chequear la conectividad.
- **Rosparam:** Permite obtener y establecer valores del servidor de parámetros desde la línea de comandos.
- **rosservice:** Muestra el tiempo de ejecución sobre un servicio, y permite imprimir mensajes que se envían al topic.
- **rosstack:** Información de la pila que se desee.

- **rostopic:** Muestra el tiempo de ejecución sobre los topics y permite imprimir mensajes que se envían a un topic.
- **rosversion:** Permite ver la versión de Ros instalada en el ordenador.

Gracias a las diversas herramientas que ofrece ROS, podemos conocer diversas características relevantes sobre nodos, topic y servicios, dentro de cada una de estas herramientas existen una serie de comandos, los cuales son de gran ayuda al momento de obtener información. Dentro de este trabajo de investigación, se utilizará con mayor frecuencia las herramientas de rostopic, rosnode y rosparam. En el caso de suscribimos a un Topic seremos capaces de obtener la información que está circulando por él, y al momento de publicar en un topic, enviamos la información a través de él, como se puede apreciar en la Tabla 2 se muestran los diferentes comandos que posee la herramienta rostopic, la cual permite conocer información más detallada acerca de un topic.

Tabla 2

Comandos de la herramienta rostopic

Comando	Descripción
1. rostopic bw	Muestra el ancho de banda consumido por topic
2. rostopic delay	retardo en la visualización del topic
3. rostopic echo	muestra los mensajes del topic por pantalla
4. rostopic find	busca topic por tipo
5. rostopic hz	Muestrea la velocidad de publicación de un topic
6. rostopic info	Muestra la información de un topic activo
7. rostopic list	Lista de topics activos
8. rostopic pub	Publica un mensaje en un tópico
9. rostopic type	Muestra un topic o un tipo de campo

Fuente: (ROS, 2018)

La herramienta de rosnode permite obtener información de los nodos activos, como se puede apreciar en la Tabla 3, los comandos de rosnode da la opción de borrar, terminar un nodo, probar la conectividad entre nodos, todo esto corresponde a los nodos que se encuentren dentro del Master de ROS.

Tabla 3

Comandos de la herramienta rosnode.

Comando	Descripción
1. rosnode info	Imprime la información acerca de un nodo
2. rosnode kill	Detiene la ejecución de un nodo
3. rosnode list	Lista de los nodos activos
4. rosnode machine	Lista de nodos ejecutándose en una máquina particular o lista de máquinas
5. rosnode ping	Prueba la conexión entre nodos
6. rosnode cleanup	borra la información de nodos que no estén activos

Fuente: (ROS, 2018)

3.3 Driver Bebop_Autonomy

El bebop_autonomy es un driver de ROS, el cual funciona para los cuadricópteros de parrot en los modelos del Bebop 1 y Bebop 2, este driver está basado en el SDK del ARDroneSDK3 el cual se encuentra en la página oficial de Parrot. Este driver fue desarrollado por el Autonomy Lab en la Universidad de Simon Fraser de Canadá, gracias a Mani Monejjemi en compañía de sus colaboradores.

Una vez ejecutado el driver este creará topics los cuales permiten la creación de suscribers para recibir la información enviada por el dron, y publishers para enviar comandos hacia el dron.

Para ejecutar el driver es necesario utilizar el comando roslaunch, debido a que este driver es un archivo .launch, la sintaxis para la escritura del comando es:

$roslaunch [carpeta que lo contiene][archivo.launch]

$roslaunch bebop_driver bebop_node.lauch

Una vez el driver sea ejecutado, genera una serie de topics los cuales permiten conocer y manejar todas las variables que posee el dron, estas pueden ser la orientación de la cámara, nivel de la batería, envió de velocidades para tele operar al dron, coordenadas del GPS, pose del dron, y la recepción de imágenes captadas por la cámara a bordo del dron.

Dentro de la ruta de instalación del driver se crean 2 carpetas las cuales son launch y config. Al momento de ejecutar el comando roslaunch con el archivo *beboo_node.lauch* lo primero que hace el driver es ejecutar el fichero de configuración *.yaml,* el cual posee todos los parámetros iniciales del dron, una vez ejecutado el driver no se podrán modificar los parámetros establecidos en este archivo, y para poder cambiarlos se tiene que proceder a la detención del nodo, modificar los valores que deseemos y volver a ejecutar el comando *roslaunch.*

Los parámetros contenidos dentro del archivo *.yaml* están relacionados con la altitud máxima y mínima que puede adquirir el dron, la distancia máxima que puede estar lejos del emisor, el ángulo máximo que puede tener el dron al momento de realizar un giro, entre otras, como se puede apreciar en la Figura 8 esta es la configuración original del archivo *.yaml* sin ninguna modificación.

```
states:
  enable_commonstate_batterystatechanged: true
  enable_commonstate_wifisignalchanged: true
  enable_overheatstate_overheatchanged: true
  enable_controllerstate_ispilotingchanged: true
  enable_mavlinkstate_mavlinkfileplayingstatechanged: true
  enable_mavlinkstate_mavlinkplayerrorstatechanged: true
  enable_flightplanstate_availabilitystatechanged: true
  enable_flightplanstate_componentstatelistchanged: true
  enable_pilotingstate_altitudechanged: true
  enable_pilotingstate_flattrimchanged: true
  enable_pilotingstate_flyingstatechanged: true
  enable_pilotingstate_navigatehomestatechanged: true
  enable_pilotingstate_positionchanged: true
  enable_pilotingstate_speedchanged: true
  enable_pilotingstate_attitudechanged: true
  enable_pilotingstate_positionchanged: true
  enable_altitudechanged: true
  enable_autotakeoffmodechanged: true
  enable_mediastreamingstate_videoenablechanged: true
  enable_camerastate_orientation: true
  enable_gpsstate_numberofsatellitechanged: true
  enable_numberofsatellitechanged: true

reset_settings: true
publish_odom_tf: true
odom_frame_id: "odom"
cmd_vel_timeout: 0.2
```

Figura 8. Archivo .yaml original

Se pueden aumentar sentencias al archivo de configuración yaml con el fin de asegurar la integridad del dron y la seguridad de las personas u operadores. A continuación se presentan las sentencias adicionales que se pueden adicionar al archivo .yaml:

- **PilotingSettingsMaxAltitudeCurrent:** Relacionada con la altitud máxima a la que se quiere mantener al dron estático, hasta que llegue un comando de avance, se encuentra en unidades de metros, el valor escogido es de 1.5 metros.
- **PilotingSettingsMaxTiltCurrent:** Relacionada con la máxima inclinación que tomará el dron el momento de desplasarse en el eje x o eje y, sus unidades están en grados, el valor escogido es de 5 grados.
- **PilotingSettingsMaxDistanceValue:** Relacionada con la distancia máxima que se puede alejar el dron del emisor que lo controla, se encuentran en unidades de metros y el valor escogido es de 10 metros.
- **PilotingSettingsNoFlyOverMaxDistaceShouldnotFlyover:** Está relacionado con la habilitación del parámetro PilotingSettingsMaxDistanceValue, se pone a 1 si queremos habilitar el parametro y 0 si queremos que el dron se aleje.
- **PilotingSettingsMinAltitudeCurrent:** Altitud minina a la que puede volar el dron, se encuentra en unidades de metros, y el valor escogido es 0 metro
- **SpeedSettingsMaxVerticalSpeedCurrent:** Maxima velocidad de ascenso o descenso, unidades entre 0 y 1, siendo 0 el minimo y 1 el máximo**.**
- **SpeedSettingsMaxRotationSpeedCurrent:** Máxima velocidad de rotación en Yaw, se encuentra en unidades de grados por segundo, el valor escogido es de 10 grados por segundo.
- **SpeedSettingsOutdoorOutdoor:** Parámetro que indica si el dron vuela en exteriores (outdoor) o en interiores (indoor), para outdoor el valor es de 1 y para indoor el valor es de 0.

3.3.1 Mensajes y Topics del driver bebop_autonomy

El driver bebop_autonomy permite acceder y utilizar diversas funciones del dron, las cuales están conformados por topics, nodos, servicios, mensajes y parámetros, este driver permite

controlar acciones como despegar, aterrizar, navegar, entre otros por medio de los topics correspondientes como son takeoff, land, cmd_vel, etc, como se presenta en la Tabla 4 se puede apreciar los tópicos antes mencionados con sus respectivos parámetros.

Tabla 4

Topics principales del driver bebop_autonomy.

Topic	Tipo de Mensaje	Descripción
Takeoff	std_msgs/Empty	Despegue vertical del suelo
Land	std_msgs/Empty	Aterrizaje de modo seguro
Reset	std_msgs/Empty	Detención de emergencia de motores
cmd_vel	geometry_msg/Twist	Control de Movimiento del Dron
camera_control	geometry_msg/Twist	Movimiento de la cámara virtual
image_raw	sensor_msgs/Image	Transmisión de video de la cámara

Fuente: (Monajjemi, 2015)

Los valores publicados en el tópico cmd_vel deben de estar en el rango de [-1 : 1], siendo 0 la mínima velocidad, y 1 o -1 la maxima velocidad, los valores de este topic son puestos a cero de manera automática al recibir un comando de mayor jerarquía como lo son takeoff, land o reset, para precautelar la integridad del dron, en la Figura 9 podemos apreciar la estructura que el tópico cmd_vel.

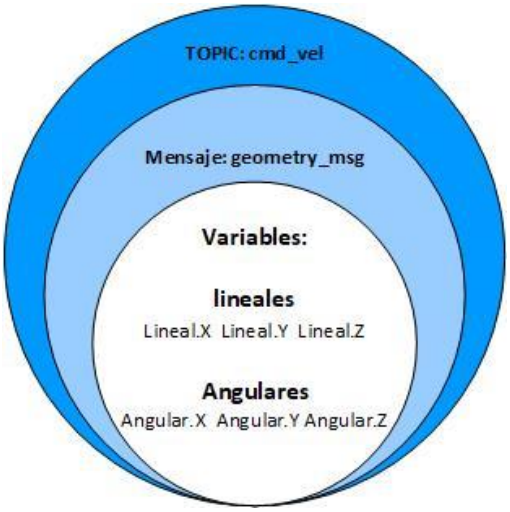


Figura 9. Estructura del topic cmd_vel

El topic cmd_vel posee 2 vectores de 3 componentes, cada uno de estos vectores indican la velocidad lineal y velocidad angular en los ejes X, Y, Z. En la Figura 10 se puede apreciar la dirección de las velocidades con las que se va a trabajar con el dron, siendo la velocidad *linear.X* equivalente a Pitch, la velocidad *Linear.Y* equivalente a Roll, la velocidad *Linear.Z* equivalente a la altura y la velocidad *Angular.Z* equivalente a Yaw.

Figura 10. Desplazamiento del Bebop 1

3.4 Robot System Toolbox

Robot System Toolbox es un toolbox de Matlab que tuvo su aparición en su versión 2015, este toolbox proporciona una interfaz entre Matlab/Simulink y ROS, la cual permite probar, verificar y simular utilizando las herramientas que ROS provee, este es compatible con la generación de código C++, lo cual permite generar un nodo de ROS a partir de Matlab. Este nodo permite trabajar con mensajes de ROS, publicar y suscribirnos a topics, acceder a servicios, entre otros, es decir permite manejar ROS desde Matlab (MathWorks, 2018).

CAPÍTULO IV

4. Desarrollo del Controlador Basado en Lógica Difusa.

4.1 Introducción

Este capítulo se enfoca en la realización de los controladores basados en lógica difusa, para lo cual primero se hace la selección de las variables que son las entradas de cada uno de los controladores difusos, posterior se realiza el tratamiento de las imágenes provenientes del dron para realizar un análisis de cada una y establecer el tipo de acción que realiza el controlador, y como parte final se realizan las diferentes reglas de control que regirán los controladores difusos. En la Figura 11 podemos apreciar de una manera general como será la configuración del sistema de control a implementar.

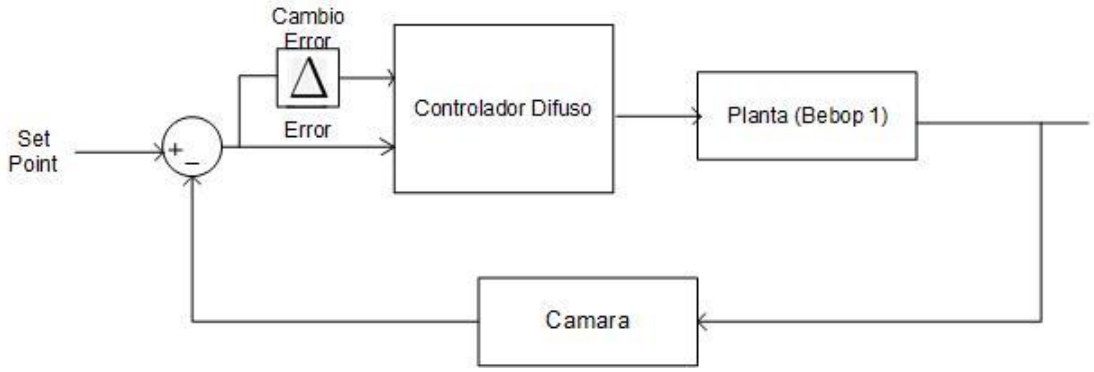


Figura 11. Sistema de Contol Difuso

4.2 Definición de Variables de Interés

Para establecer las variables que se utilizan como entradas de los controladores basados en lógica difusa, es necesario conocer los diferentes datos a los se puede acceder en el dron, y seleccionar aquellos que estén ligados al movimiento del dron. Como se apreció en el capítulo anterior en la sección 3.3.1 el tópico ligado con el movimiento del dron es el topic cmd_vel, dentro de este tópico tenemos acceso a las velocidades lineales en X, Y, Z y una velocidad angular en Z.

En el driver Bebop_autonomy se expone que las variables de velocidad *Linear.X* está relacionada con el ángulo Ptich, *Linear.Y* está relacionada con el ángulo Roll, *Linear.Z* está relacionada con la altura del dron y la velocidad *Angular.Z* está relacionada con el ángulo Yaw, en la Figura 12 se puede apreciar los movimientos del dron, ligados a cada uno de sus ángulos.

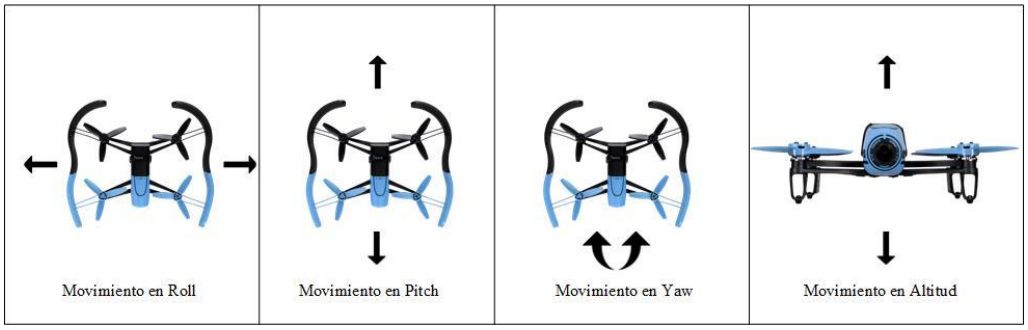


Figura 12. Movimientos del Dron Bebop 1

El seguimiento de la trayectoria del dron se realizó con el uso de las variables de velocidad Linear.X, Linear.Y y Angular.Z, cada una de estas variables se escogió en base a la respuesta que genera su uso en el dron, dando por resultado:

- Linear.X para que el dron realice un movimiento hacia adelante o hacia atrás.
- Linear.Y para que el dron realice un movimiento lateral ya sea a la izquierda o derecha.
- Angular.Z para que el dron realice un movimiento sobre su propio eje.

Tomando en cuenta que el dron va a permanecer a una altura constante se prescindió de la variable Linear.Z, la cual se encarga del movimiento ascendente y descendente del dron.

4.2.1 Dispositivo de Visión

Para la realización de este proyecto se analizaron diversas opciones de sensores como fueron sensor GPS integrado en el dron, y la odometria que este ofrecía, no obstante el tiempo de muestreo de estos datos es de 5 Hz, lo cual para fines del proyecto es muy lento para poder realizar una acción de control, por este motivo fue necesario utilizar la cámara integrada que posee el dron, la misma está montada en la parte delantera del dron, tiene la característica de ser una cámara ojo de

pez, la cual cuenta con aproximadamente 180º de visión, posee un sensor de 14 megapixeles, con una resolución en video de 1920 x 1080 pixeles a 30 fps o frames per second, con una codificación H.264(MPEG4), y su resolución en fotografía es de 4096 x 3072 pixeles con codificaciones en formatos JPEG, RAW y DNG (Parrot, 2018).

Dentro del dron vienen incorporados algoritmos para la estabilización de imagen y video los cuales son propios del dron, estos algoritmos permiten tener imágenes de buena calidad y sin distorsiones que se pueden producir por las condiciones habituales de vuelo, como pueden ser turbulencias, giros bruscos (Salcedo, 2018).

Gracias a estos algoritmos que ofrecen una estabilización de la imagen en los 3 ejes, es posible manipular por software la inclinación de la cámara, haciendo uso del driver bebop autonomy, con un mensaje de tipo cmd_vel/Twist, y accediendo al tópico camera_control. El Angulo de apertura de imagen que se presenta para la cámara del dron es de aproximadamente 50º en el eje vertical y 80º en el eje horizontal.

Una vez conectados con el driver de bebop autonomy, podemos suscribirnos al tópico de image_raw, con el fin de obtener el flujo de fotogramas que conforman el video que trasmite el dron hacia el ordenador, estos fotogramas se almacenaran como mensajes de tipo sensor_msgs/Image, el cual es un mensaje típico de ROS para codificar datos de imágenes, la resolución de las imágenes trasmitidas será de 640 x 368 pixeles, a una tasa de 30 fps, en la Figura 13 se puede apreciar los datos recibidos desde el tópico image_raw.

Property	Value
MessageType	'sensor_msgs/Image'
Header	*1x1 Header*
Height	368
Width	640
Encoding	'rgb8'
IsBigendian	0
Step	1920
Data	*706560x1 uint8*

Figura 13. Datos del topic image_raw

La codificación con la cual el dron bebop envía las imágenes es RGB8, cada valor de la imagen se encuentra en la variable Data, esta variable tiene un valor de 706560 pixeles. La resolución de la imagen es de 640 x 368 lo cual da un total de 235520 pixeles, por lo tanto, si multiplicamos el número total de pixeles de cada imagen por tres, el cual será cada una de las componentes RGB, tendremos un valor de 706560, lo que nos dice que el valor de cada una de las componentes de RGB se encuentra en esta variable y es necesario reordenar los pixeles.

Siguiendo el procedimiento que sugiere (Lavid, 2017) en su trabajo, se procede a realizar el ordenamiento de cada uno de los pixeles para formar la imagen, pero solo se ordenan los pixeles correspondientes a la componente G de la imagen, esto se realiza con el fin de reducir tiempo de cómputo para el ordenador, ya que la respuesta del dron tiene que ser casi en tiempo real, y para agilitar los análisis que se realizarán a la imagen. Después de realizar el ordenamiento de los pixeles tendremos la imagen con la resolución de 640 x 320 pixeles, en una escala de grises como se puede apreciar en la Figura 14.

Figura 14. Imagen recibida desde Bebop 1

4.2.2 Análisis de Imagen

Después de realizar el ordenamiento de los pixeles para formar la imagen en escala de grises, se procede a realizar el análisis de las imágenes provenientes del dron, con la finalidad de que las imágenes aporten la información necesaria para la realización del proyecto.

Al momento de trabajar con imágenes y al realizar una extracción de características de esta, se tiene algunos problemas con respecto a la influencia de la luz, ya que esta tiende a iluminar u obscurecer ciertas zonas del objeto, por esta razón se decidió trabajar con una detección de rectángulos de color blanco, ya que al trabajar con una imagen en escala de grises el color blanco es más fácil de detectar.

Para formar los diferentes rectángulos que servirán para la detección, se realizaron diversas pruebas con diferentes materiales, como cartón, espuma Flex y madera. Se decidió optar por la utilización de madera MDF, ya que esta presenta una mejor resistencia al agua, no tiende a moverse con fuertes ventiscas de aire. Las piezas de madera cuentan con un grosor de 1.2 milímetros, las dimensiones son de 1 metro de largo por 30 centímetro de ancho, se puede apreciar en la Figura 15 una de las ocho piezas construidas para fines del proyecto.

Figura 15. Piezas de Madera para Fomar Trayectoria

El siguiente paso del análisis de la imagen es realizar la binarización de la imagen obtenida, para obtener una nueva imagen solo en blanco y negro, para realizar esto se recurrió al uso del Image Processing Toolbox de Matlab, con la ayuda del comando im2bw, este comando convierte una imagen en escala de grises a una imagen binaria, al reemplazar todos los píxeles de la imagen con el valor de 1 para blanco y por el valor de 0 para negro, para realizar la conversión se basa en un nivel de umbral o threshold, si el valor del pixel está por encima de nivel de umbral lo reemplaza por el valor de 1, y si está por debajo del nivel lo reemplaza por el valor de cero, como se puede apreciar en la Figura 16, tenemos la imagen en escala de grises obtenida con el dron, y la imagen binarizada con un valor de umbral de 0.95.

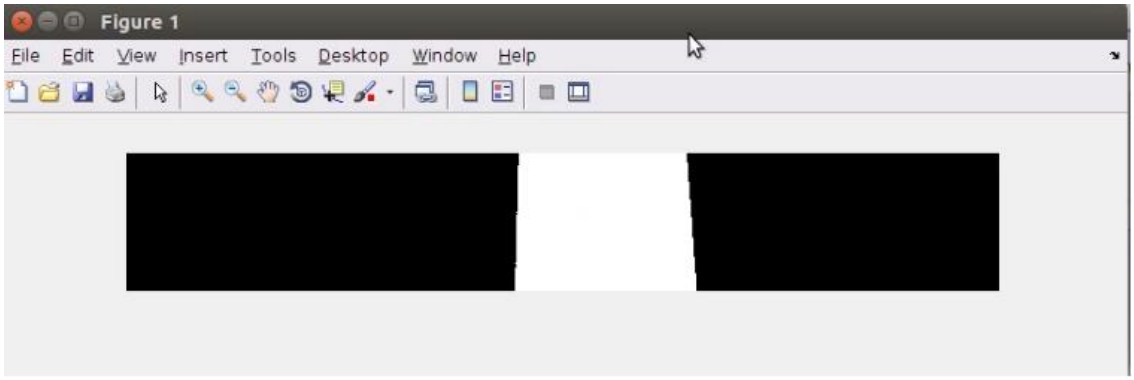


Figura 16. Imagen Binarizada obtenida del dron

Después de tener la imagen binarizada se le aplico un filtro de mediana con la ayuda del comando medfilt2, este comando realiza un filtro de mediana a la imagen con el fin de eliminar ruido. Después de aplicar el filtro de mediana en la imagen, se procede a realizar el cálculo del valor de la pendiente que tiene la pieza de madera, definiendo por pendiente a la inclinación que tendrá la figura de madera con respecto al eje de las absisas. Para realizar este cálculo se utilizó el comando find de Matlab, el cual retorna la posición en fila y columna de los valores diferentes de cero dentro de una matriz, después de almacenar estos valores procedemos a introducirlos en el comando polyfit, este comando realiza el ajuste de una curva polinómica, utilizando los puntos de datos a través de cálculos de mínimos cuadrado, para obtener la pendiente es necesario especificar que el polinomio de ajuste sea de grado uno, a continuación, en la Figura 17 se muestra como se realiza el obtiene la pendiente de la figura detectada.

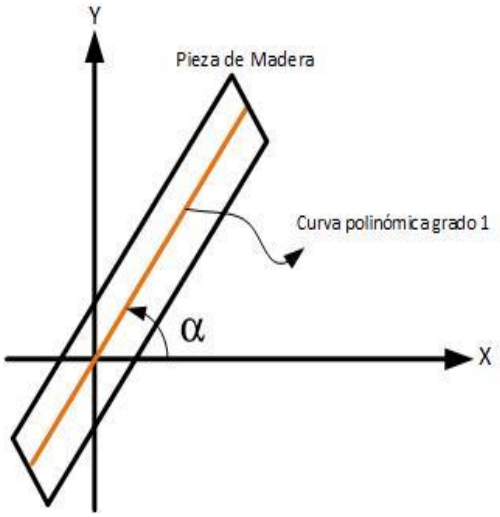


Figura 17. Obtención de la pendiente de la pieza de madera

Continuando con el análisis de las imágenes, otra variable que es importante conocer es la del centroide, ya que con esta podemos conocer el centro de masa del objeto que estemos analizando, en este caso obtendremos el centro de las figuras de madera que formarán un camino por el cual el dron deberá seguir. Para obtener el valor del centroide procedemos a utilizar el comando

regionprops de Matlab, este comando mide diversas propiedades de una sección de imagen, como pueden ser: área, perímetro, centroide, convecciones, numero de Euler, entre otras, para nuestro fin solo utilizaremos la medida del centroide. Como se puede apreciar en la Figura 18 solo nos muestra el centroide de la figura de madera detectada.

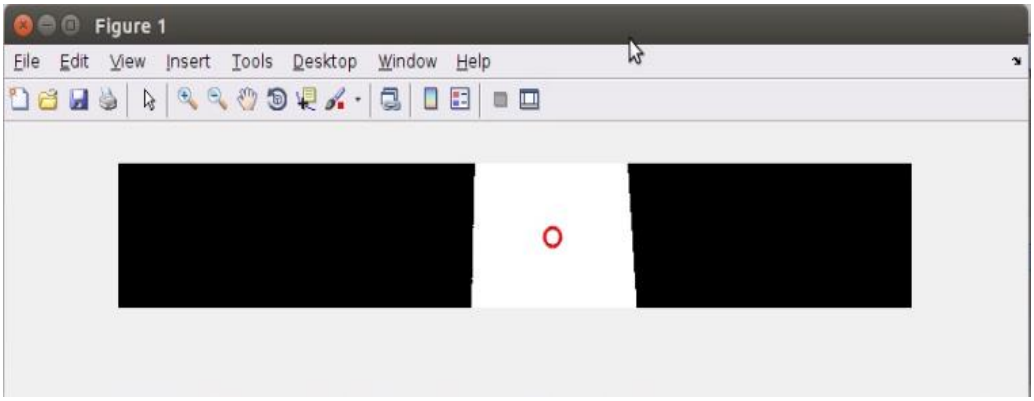
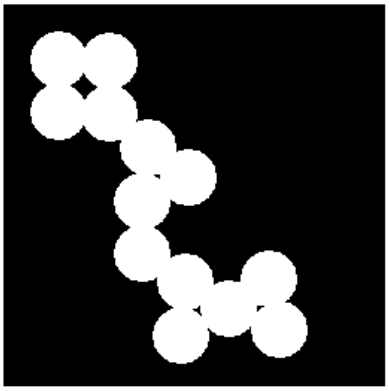


Figura 18. Deteccion del centroide en la pieza de madera

La última variable necesaria para continuar con el análisis de la imagen es la obtención del área de la figura detectada, definiéndose por área al valor de la superficie formada únicamente por los pixeles con valor de 1 en la imagen binarizada. Para obtener este valor utilizamos el comando de Matlab *bwarea*, el cual nos permite realizar una estimación al cálculo del área de cualquier objeto dentro de una imagen binarizada. En la Figura 19 se puede apreciar el cálculo del área a una figura binarizada.



```
bwarea(BW)

ans = 1.4187e+04
```

Figura 19. Área de figura detectada
Fuente: (MathWorks, 2018)

4.3 Desarrollo de los Controladores Difusos

Para el desarrollo de los controladores difusos que realizarán las acciones de control en el dron, se van a tomar en cuenta las características obtenidas de las imágenes como son la pendiente y el centroide de la figura detectada. Las variables de entrada de cada controlador difuso serán el error y el cambio de error de la pendiente y de la posición del centroide, y como salida obtendremos el valor de velocidad que se enviará por el tópico cmd_vel.

4.3.1 Controlador Difuso para Yaw

Al valor de la pendiente se lo asigno a la variable de movimiento yaw, esto se realizó con el fin de que, al momento de detectar alguna inclinación de la pieza de madera, el dron este en la capacidad de girar sobre su propio eje, y dejar a la pieza de madera con una orientación vertical, para que de esta manera el dron pueda seguir en una línea recta.

Los datos máximos de entrada para el controlador difuso fueron obtenidos de manera experimental, colocando al dron a una altura de 80 centímetros y haciéndolo girar en sentido horario y sentido anti horario, dando por resultado valores máximos aproximados a -0.25

pixeles/pixeles cuando se giró el dron en sentido anti horario y a un ángulo de giro de aproximadamente 45º y un valor de 0.25 pixeles/pixeles cuando se giró al dron en sentido horario y a un ángulo de giro de aproximadamente 45º. En la Figura 20, Figura 21 se puede apreciar el valor de pendiente obtenido para cada sentido de giro del dron.

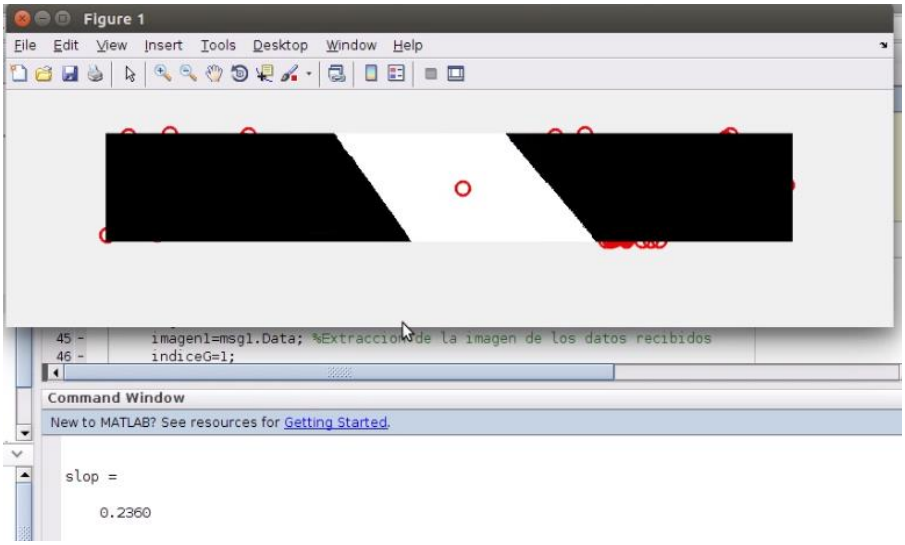


Figura 20. Valor de Pendiente en sentido Horario

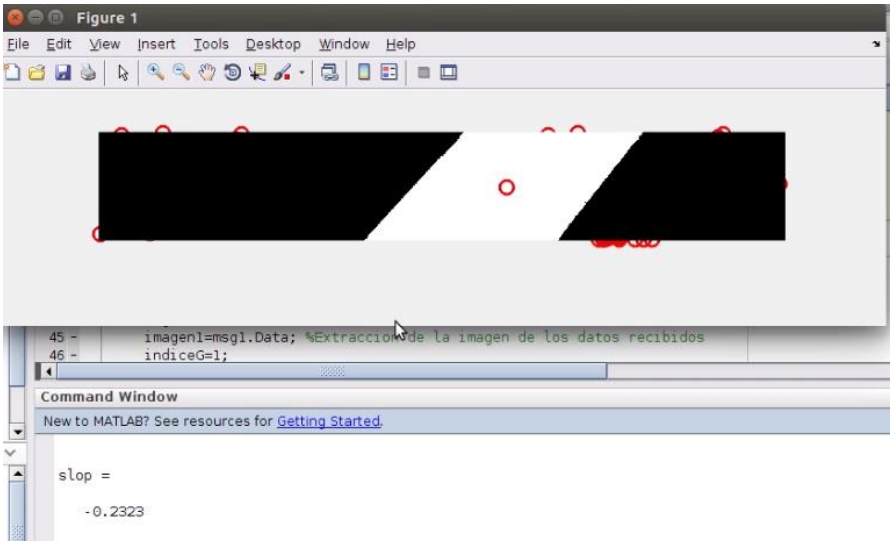


Figura 21. Valor de Pendiente en Sentido Antihorario

Para el controlador difuso se optó por un rango de fusificación que vaya desde -0.25 pixeles/pixeles hasta 0.25 pixeles/pixeles tanto para el error, así como para el cambio del error,

teniendo como valor deseado o set point el valor de cero, el controlador también está compuesto por 5 funciones de membrecía triangulares las cuales van nombradas de a siguiente manera: GN (grande negativo), PN (pequeño negativo), Z (cero), PP (pequeño positivo) y GP (grande Positivo), y el rango del valor de salida comprendido entre -0.3 unidades a 0.3 unidades según driver bebop autonomy. En la Tabla 5 se pueden apreciar las reglas que conforman al controlador difuso.

Tabla 5

Controlador Fuzzy Para Yaw

		Error				
		GN	**PN**	**Z**	**PP**	**GP**
Cambio de Error	**GN**	GP	GP	GP	PP	Z
	PN	GP	GP	PP	Z	PN
	Z	GP	PP	Z	PN	GN
	PP	PP	Z	PN	GN	GN
	GP	Z	PN	GN	GN	GN

4.3.2 Controlador difuso para Roll

Para el desarrollo del controlador difuso de roll, se utilizó la variable de posición del centroide, esto se lo realizó con la finalidad de mantener la pieza de madera detectada centrada, y que al momento de detectar que el valor del centroide se aleja le centro de la imagen realice movimientos laterales para que el centroide de la figura vuelva al centro de la imagen.

Los datos para la entrada del controlador difuso se los tomó del ancho de la imagen que se recibe del dron, la imagen posee un ancho de 640 pixeles, por lo tanto, el centro de la imagen es 320 pixeles y el rango de entrada para el controlador difuso en la variable del error es de -320 a 320 pixeles, y el rango de entrada para la variable del cambio de error también será de -320 a 320 pixeles, también se definieron para este caso 5 funciones de membrecía triangulares las cuales están nombradas de la siguiente manera: GN (grande negativo), PN (pequeño negativo), Z (cero), PP

(pequeño positivo) y GP (grande Positivo), y el rango del valor de salida comprendido entre -0.35 unidades a 0.35 unidades según driver bebop autonomy. En la Tabla 6 se pueden apreciar las reglas que conforman al controlador difuso.

Tabla 6
Controlador Fuzzy para Roll

		Error				
		GN	**PN**	**Z**	**PP**	**GP**
Cambio de Error	**GN**	GP	GP	GP	PP	Z
	PN	GP	GP	PP	Z	PN
	Z	GP	PP	Z	PN	GN
	PP	PP	Z	PN	GN	GN
	GP	Z	PN	GN	GN	GN

4.4 Implementación de los Controladores Difusos

En la etapa de implementación de los controladores se utilizó la herramienta *Fuzzy Logic Designer* de Matlab en su versión 2016b, esta herramienta permite diseñar y probar sistemas de inferencia para realizar el modelamiento de sistemas complejos, podemos diseñar los sistemas de inferencia de mandami y sugeno, agregar o eliminar variables de entrada y salida, especificar las funciones de membrecía, definir las diferentes reglas de control, observar los mapas de superficie de salida para los diferentes sistemas de inferencia (MathWorks, 2018).

Para la implementación de los controladores difusos, se utilizó el sistema de inferencia de Mamdani por tener las características de ser fácil de comprender, es más simple para formular las reglas de control y es ampliamente utilizado, a continuación, en la Figura 22 se puede apreciar la pantalla principal de la herramienta *Fuzzy Logic Designer*, en la cual tenemos una vista previa de cómo está configurado el controlador difuso.

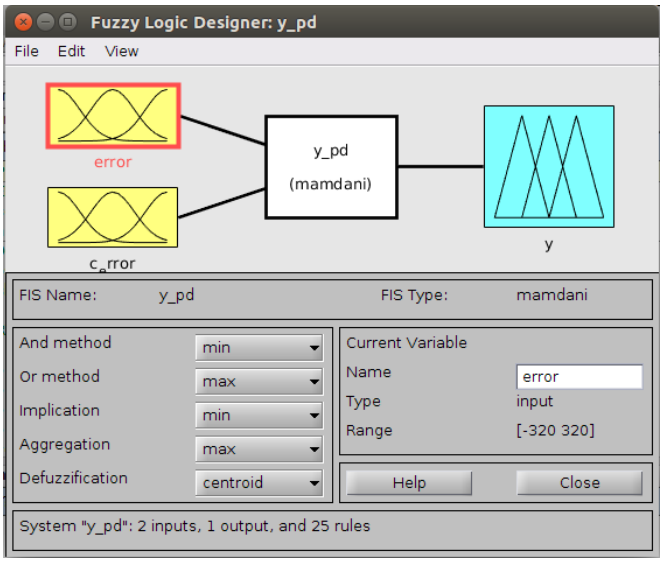


Figura 22. Pantalla Principal de Fuzzylogicdesigner

Una de las principales funciones que presenta esta herramienta es la de simular el controlador difuso que se ha desarrollado, en este caso podemos variar el valor de las entradas del controlador y apreciar cual será el valor de salida, de esta forma podemos constatar su correcto funcionamiento, como podemos apreciar en la Figura 23 simulamos el funcionamiento del controlador desarrollado para la variable Roll para constatar que las reglas de control estén bien definidas.

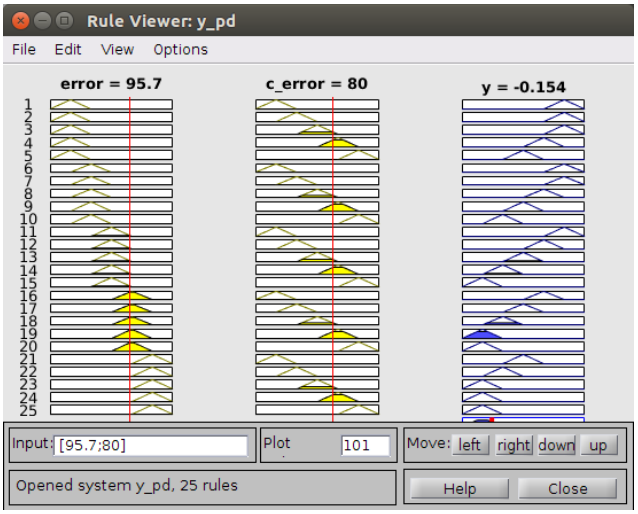


Figura 23. Reglas del Controlador Difuzo para Roll

Otra de las funciones que presenta esta herramienta es la de observar la superficie de control, esta muestra la gráfica correspondiente a la entrada vs salida del controlador, en nuestro caso al poseer 2 entradas y una salida mostrará una superficie, en esta podemos apreciar de una forma gráfica el valor que tomará la salida para los valores que tenga la entrada, en la Figura 24 podemos apreciar la superficie de control para el controlador difuso de roll.

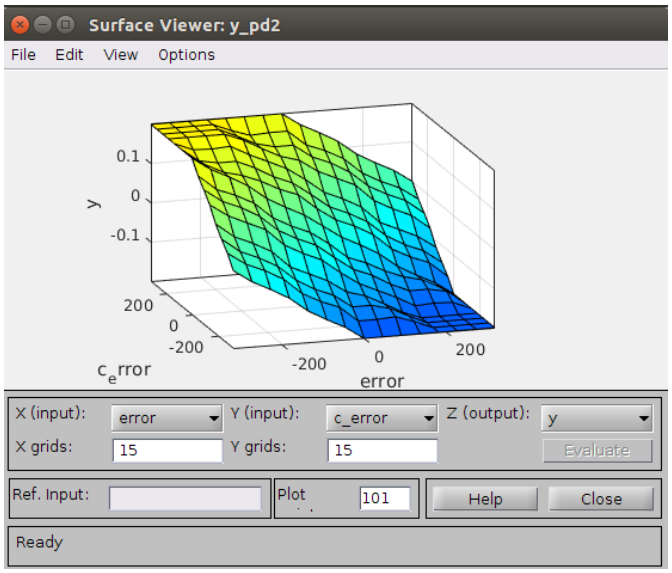


Figura 24. Supeficie de Control para Controlador Difuso de Roll

4.5 Diseño del algoritmo de control

Para el diseño del algoritmo de control que regirá sobre el seguidor de trayectoria, es necesario conocer el camino que seguirán los diferentes datos que provengan tanto del dron como del ordenador, esto servirá para tener una idea más clara sobre la comunicación que se realiza entre el dron, ROS y el ordenador, como se puede apreciar en la Figura 25 tenemos el diagrama de flujo que representa el camino que sigue la información que se envía desde el dron al ordenador y viceversa.

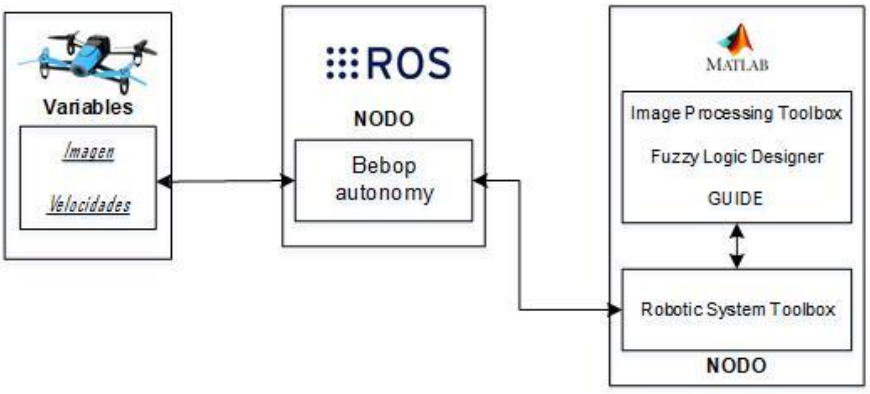


Figura 25. Diagrama de flujo de la información

El flujo de la información empieza desde el dron el cual envía las diversas imágenes captadas por su cámara frontal, estas pasan a ser enviadas de manera inalámbrica al ordenador, en el ordenador el nodo bebop autonomy se encarga de guardar la información de las imágenes en el topic image_raw, desde Matlab se subcribe a este topic y procedemos a realizar el análisis de cada imagen.

Una vez realizado el análisis de las imágenes y establecida la acción de control, Matlab publica en el tópico cmd_vel las velocidades resultantes de los controladores, estas son enviadas al nodo bebop autonomy, el cual envía los valores de las velocidades al dron para que realice los diferentes movimientos que garantizan el seguimiento de la trayectoria, formando así un sistema cerrado en el flujo de la información.

En la Figura 26 se muestra el diagrama de flujo correspondiente al algoritmo de seguimiento de trayectoria, el cual consiste en la inicialización de los nodos de ROS en el ordenador como en Matlab, la inicialización de variables, extracción de características de las imágenes y por último las acciones de control provenientes de los controladores difusos.

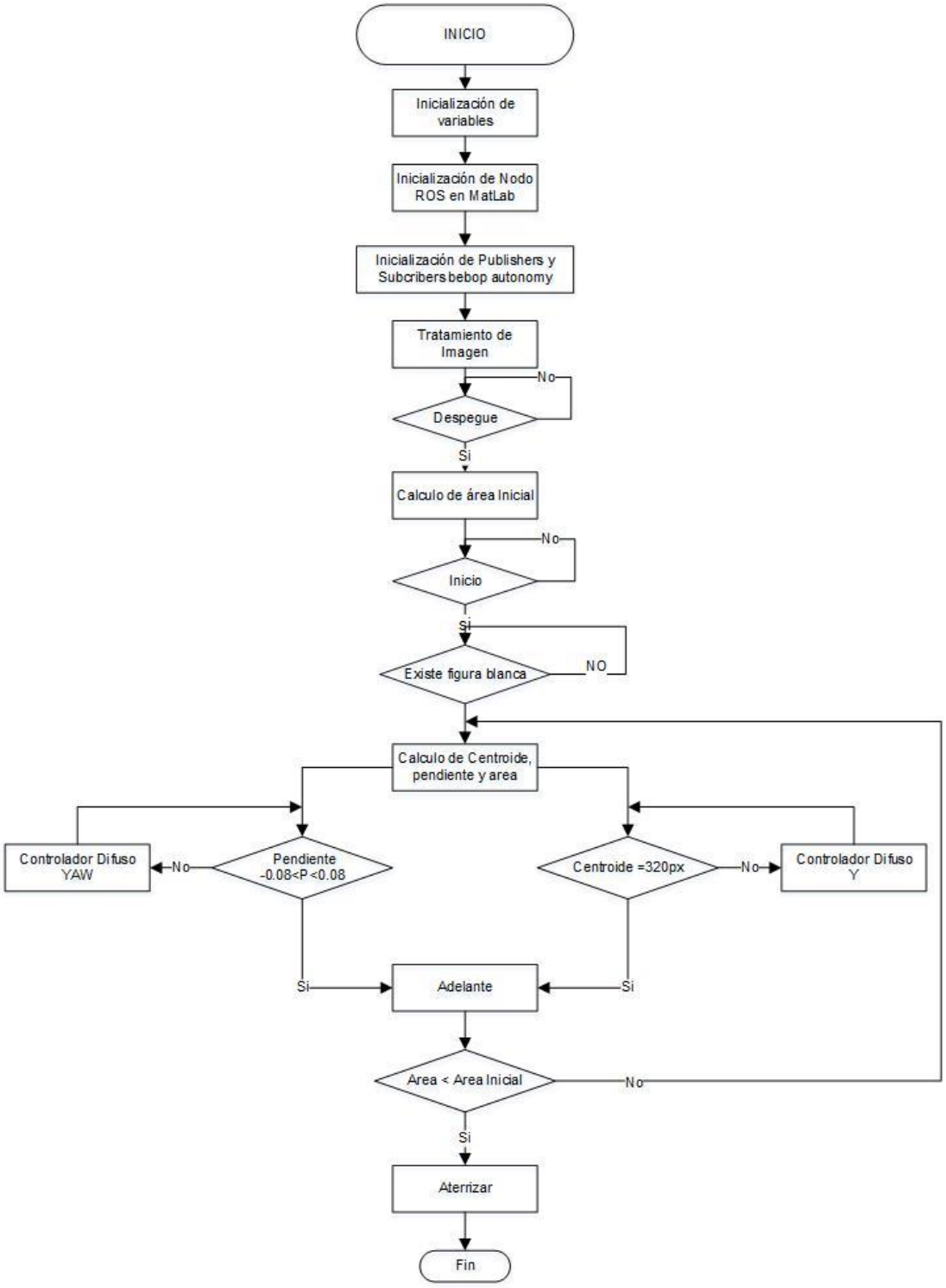


Figura 26. Diagrama de flujo del algoritmo de seguimiento de trayectoria

A continuación, se detallará el código realizado para cumplir cada una de las actividades mencionadas en el diagrama de flujo, como se puede apreciar en la Tabla 7 tenemos el código correspondiente a la inicialización de las variables que se utilizaron en el programa de seguimiento de trayectorias.

Tabla 7

Código correspondiente a la inicalización de variables

```
close all;
clear all;
rosshutdown; %Cierre de conexiones de ROS
%-------------------------------------------------------
%% Inicializacion de variables
%-------------------------------------------------------
global start;
global land;
global takeOff;
global velconst;
global adelante;
global atras;
global izq;
global der;
cerrory=[0 0]; % vector cambio de error Y
cerroryaw=[0 0]; % vector cambio de error YAW
pos=[0 0]; % vector para guardar posicion
xp=0; % vector de valores de posicion en x
yp=0; % vector de valores de posicion en Y
yc=0; % vector de valores de centroide
yawc=0; % vector de valores de pendoiente
o=1; % valor para desplazamiento de valores de odometro
c=1; % valor para desplazamiento de valores de posicion centroide y pendiente
```

Después de realizar la inicialización de variables, se procede a iniciar el nodo ROS de Matlab para lograrlo se utiliza el comando rosinit, después es necesario inicializar los publicadores como son el de la cámara, para el despegue, aterrizaje y las velocidades, cada uno con sus respectivos mensajes. A continuación de inicializan los subscriptores de los cuales vamos a recibir información como son los de imagen y odometría. En la Tabla 8 se puede apreciar el código de la inicialización de publicadores y subscriptores.

Tabla 8

Código de inicialización de publicadores y subscriptores

```
rosinit; %inicialización de ROS en Matlab

%%%%%% Tópicos a utilizar %%%%%%%%%%
% /bebop/camera_control --- tópico de control de cámara digital
% /bebop/land           --- tópico para aterrizar
% /bebop/takeoff        --- tópico para despegar
% /bebop/cmd_vel        --- Tópico para enviar velocidades
% /bebop/image_raw      --- Tópico para recibir imagen de cámara
% /bebop/odom           --- Tópico para recibir valores de odometría
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Inicialización de cámara
pubCamera=rospublisher('/bebop/camera_control','geometry_msgs/Twist');
camera_control_msgs=rosmessage(pubCamera);
camera_control_msgs.Angular.Y=-80;
camera_control_msgs.Angular.Z=0;
send(pubCamera,camera_control_msgs);
% inicializacion de publishers: aterrizaje, despegue, velocidades
pubLand=rospublisher('/bebop/land','std_msgs/Empty');
land_msgs=rosmessage(pubLand); %mensaje para aterrizar
pubTakeoff=rospublisher('/bebop/takeoff','std_msgs/Empty');
takeoff_msgs=rosmessage(pubTakeoff); %mensaje para despegar
pub=rospublisher('/bebop/cmd_vel','geometry_msgs/Twist');
cmd_vel_msgs=rosmessage(pub); %mensaje para velocidades
% inicializacion de subcribers imagen de dron y odometria
sub = rossubscriber('/bebop/image_raw'); % subcriptor para imagen de dron
odom = rossubscriber('/bebop/odom'); %subscriptor para odometria
pause(1);
r=robotics.Rate(20); %Pediodo de conexion con ROS
run('HMI_Tesis3.m') %Ejecucion de in terfas grafica
```

A continuación, se realiza el tratamiento a la imagen que el dron nos envía, para esto se tiene que realizar el ordenamiento de pixeles con el fin de obtener una imagen con la cual se pueda trabajar. Después se realiza la binarización de la imagen para poder realizar la identificación del color blanco. En la Tabla 9 se puede apreciar el código realizado para esta sección del programa.

Tabla 9

Código correspondiente al tratamiento de imagen

```
while(1)
%% Recepcion de imagen
   centro=320; %centro de imagen deseado
   pendiente=0; % valor de pendente deseado
   msg1 = receive(sub,20); %Recepcion de datos desde ROS
   imagen1=msg1.Data; %Extraccion de la imagen de los datos recibidos
   indiceG=1;
   fila=1;
   col=1;
   indice1=1;
   %Extraccion de los pixeles que forman la componente G
   for indice0 = 2:3:706559
      imagenG(indiceG,1)=imagen1(indice0,1);
      indiceG=indiceG+1;
   end
   %Formacion de la componente G de la imagen
   for fila = 1:1:368
      for col = 1:1:640
         imagenG1(fila,col)=imagenG(indice1,1);
         indice1=indice1+1;
      end
   end
   subplot(2,1,1),imshow(imagenG1);title('Imageb Recibida');
   B1=imagenG1;
   %A=B1(184:368, :); % 148*640
   A=B1(268:368, :); % 100*640
   Z=im2double(B1); %Componente G de la imagen RGB
   B=im2bw(A,0.95);
   %B=im2bw(A,0.65);
   % filtro de media para eliminar ruido
   C=medfilt2(B,[3,3]);
   subplot(2,1,2),imshow(C);title('Imagen B/N')
```

Una vez realizado el análisis de la imagen, procedemos a realizar las sentencias correspondientes para el despegue del dron. Una vez realizado el despegue se procede a realizar el cálculo del área inicial, durante el transcurso de las pruebas de funcionamiento se pudo notar que al momento de realizar el despegue el dron tarda un tiempo en estabilizarse, por lo que el cálculo del valor del área puede verse afectado. Es por esta razón que se decidió dejar un valor de área inicial fijo, este valor se obtuvo realizando diversas mediciones y realizando un promedio de todos los valores de área

obtenidos a una altura de 80 centímetros del piso. En la Tabla 10 se puede apreciar el código para el despegue y para mantener al drone en una posición estática.

Tabla 10

Código correspondiente al despegue

```
if takeOff==1
        cmd_vel_msgs.Linear.X=0;
        cmd_vel_msgs.Linear.Y=0;
        cmd_vel_msgs.Linear.Z=0;
        cmd_vel_msgs.Angular.Z=0;
        send(pub,cmd_vel_msgs);
        send(pubTakeoff,takeoff_msgs);
        pause(3);
      %cálculo de area inicial
        %area1 = bwarea(C);
        area1= 1.4533e+04; % imagen de 100*640
        takeOff=0;
        velconst=1;
    end
    if velconst == 1
        cmd_vel_msgs.Linear.X=0;
        cmd_vel_msgs.Linear.Y=0;
        cmd_vel_msgs.Linear.Z=0;
        cmd_vel_msgs.Angular.Z=0;
        send(pub,cmd_vel_msgs);
    end
```

Después de ejecutar el código de despegue, el dron se quedará en una posición estática hasta recibir el comando de inicio de la aplicación. En el código de inicio de la aplicación se procede a realizar la detección de la figura en color blanco, y una vez detectada la figura se realiza el cálculo del área la cual nos servirá para realizar el aterrizaje cuando ya no detecte la figura. También se realiza el cálculo del centroide y de la pendiente con el fin de utilizar los valores como entradas para los controladores difusos. En la Tabla 11 podemos apreciar el código utilizado para realizar la detección de la figura y cálculo del área, pendiente y centroide.

Tabla 11

Código para la detección de la figura y cálculo del área, pendiente y centroide

```
      if start==1
        velconts=0;
    %   propiedades de imagen C
        stats = regionprops(C);
        %Cálculo de centroide de línea blanca
        Centroides0 = regionprops(C,'Centroid', 'Area', 'PixelIdxList');
    %   detección de rectángulo blanco
        hold on;
        for i = 1:numel(stats)
            for j=1:1:size(Centroides0)
                rectangle('Position', stats(i).BoundingBox, ...
                'Linewidth', 3, 'EdgeColor', 'g', 'LineStyle', '-');
                hold on;
plot(Centroides0(j).Centroid(1,1),Centroides0(j).Centroid(1,2),'or','LineWidth',2,'MarkerSize',10);
%pinta circulo alrededor del centroide
                ycentro=(Centroides0(j).Centroid(1,1));%valor del centroide en Y
                %error y cambio de error  en Y
                yc(c)=ycentro;
                errory= centro-ycentro
                if cerrory(2) ~= errory
                    cerrory(1)=cerrory(2);
                    cerrory(2)= errory;
                end
                derrory= diff(cerrory) % PD
                %derrory= cerrory(1)+cerrory(2) %PI
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                % calculo pendiente
                [y, x] = find(C);
                coeff = polyfit(x,y,1); % ajuste polinomial de grado 1
                slo = coeff(1); % pendiente del la figura detectada
                intercept = coeff(2);
                yawc(c)=slo;
                c=c+1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
               % error y cambio de error en yaw
                erroryaw= pendiente-slo;
                if cerroryaw(2) ~= erroryaw
                    cerroryaw(1)=cerroryaw(2);
                    cerroryaw(2)= erroryaw;
                end
                derroryaw= diff(cerroryaw);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %diferencia de Areas
                area2 = bwarea(C);
                diffarea=area2-area1;
```

Una vez realizado el cálculo de la pendiente y el centroide, procedemos a utilizar esos datos para calcular el error y cambio de error de estas variables. Después utilizamos estos valores para

las entradas de los controladores difusos. En la Tabla 12 se aprecia el código correspondiente a la evaluación de los controladores difusos.

Tabla 12

Código correspondiente para la evaluación del los contrladores fuzzy

```
            % Controlador con error y cambio de error
            % evalfis([error cambio_error], controlador);
            yvel=evalfis([errory derrory],posy);
            %yawvel= evalfis([erroryaw derroryaw],posyaw); % yaw PD
            yawvel= evalfis(slo,posyaw);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%

            if slo>-0.08 && slo<0.08 %movimiento para adelante
              cmd_vel_msgs.Linear.X=0.3;
              cmd_vel_msgs.Linear.Y= -(0.4*yvel);
              if cmd_vel_msgs.Linear.X < 0
                  cmd_vel_msgs.Linear.X = 0;
              end

              if cmd_vel_msgs.Linear.X > 0.3
                  cmd_vel_msgs.Linear.X = 0.3;
              end
              cmd_vel_msgs.Linear.Z=0;
              send(pub,cmd_vel_msgs);
              disp('adelante')
            end

            if cmd_vel_msgs.Linear.X < 0
                  cmd_vel_msgs.Linear.X = 0;
            end
            cmd_vel_msgs.Linear.Y= -(0.4*yvel);
            cmd_vel_msgs.Linear.Z=0;
            cmd_vel_msgs.Angular.Z= (0.4*yawvel); % para yaw proporcional
            send(pub,cmd_vel_msgs);
        end
    end
```

Para finalizar, se realiza la comparación entre el área inicial y el área medida, si el valor de la diferencia decae por debajo de cero, eso quiere decir que ya no existe camino y por ende el dron aterrizará. También se implementaron sentencias que permitan el control manual del dron con el

fin de poder ubicarlo en una posición que facilite la detección en caso de existir ráfagas de viento o un mal despegue. En la Tabla 13 se presenta el código correspondiente al aterrizaje y control manual.

Tabla 13

Código correspondeinte al aterrizaje y control manual

```
    % chequea si no existe linea entonces aterriza
            elseif (round(area2-area1)< 0)
                cmd_vel_msgs.Linear.X=0; cmd_vel_msgs.Linear.Y=0;
                cmd_vel_msgs.Linear.Z=0; cmd_vel_msgs.Angular.Z=0;
                send(pub,cmd_vel_msgs);
                send(pubLand,land_msgs);
                disp('aterrizar')
          send(pub,cmd_vel_msgs);
       end
       if land == 1
          start=0;
          cmd_vel_msgs.Linear.X=0; cmd_vel_msgs.Linear.Y=0;
          cmd_vel_msgs.Linear.Z=0; cmd_vel_msgs.Angular.Z=0;
          send(pub,cmd_vel_msgs);
          send(pubLand,land_msgs); land = 0;
       end
        if adelante == 1
          cmd_vel_msgs.Linear.X= 0.4; cmd_vel_msgs.Linear.Y=0;
          cmd_vel_msgs.Linear.Z=0; cmd_vel_msgs.Angular.Z=0;
          send(pub,cmd_vel_msgs); adelante=0;
       end
      if atras == 1
          cmd_vel_msgs.Linear.X= -0.4; cmd_vel_msgs.Linear.Y=0;
          cmd_vel_msgs.Linear.Z=0; cmd_vel_msgs.Angular.Z=0;
          send(pub,cmd_vel_msgs); atras=0;
      end
      if izq == 1
          cmd_vel_msgs.Linear.X=0; cmd_vel_msgs.Linear.Y= 0.4;
          cmd_vel_msgs.Linear.Z=0; cmd_vel_msgs.Angular.Z=0;
          send(pub,cmd_vel_msgs); izq=0;
      end
       if der == 1
          cmd_vel_msgs.Linear.X=0; cmd_vel_msgs.Linear.Y= -0.4;
          cmd_vel_msgs.Linear.Z=0;cmd_vel_msgs.Angular.Z=0;
          send(pub,cmd_vel_msgs); der=0;
       end
```

CAPÍTULO V

5. Sistema de Detección y Seguimiento de Trayectoria

5.1. Introducción

El sistema desarrollado para el seguimiento de la trayectoria, definida por el usuario está compuesto por el dron, el ordenador portátil y las figuras de madera, las cuales forman la trayectoria a seguir, como se puede apreciar en la Figura 27 tenemos una visión general de cómo está compuesto este sistema en el cual la comunicación se la realiza mediante WiFi.

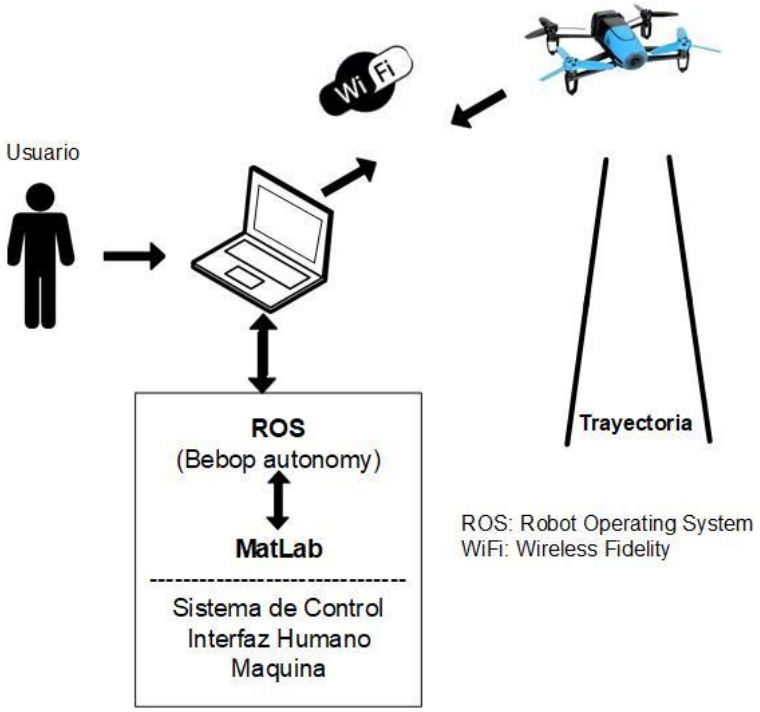


Figura 27. Sistema de Seguimiento de Trayectoria

En el sistema de seguimiento de trayectoria, la parte fundamental es el ordenador, debido a que este es el encargado de establecer la comunicación con el dron y a su vez realizar las acciones de control correspondientes para que el sistema funcione, para garantizar el correcto funcionamiento

del sistema se tienen requerimientos a nivel de hardware y software, siendo el requerimiento de software el más relevante.

Para la parte de software se necesitan los siguientes programas:

- Sistema operativo Ubuntu 16.04.
- ROS (Robot Operating System) versión Kinetic.
- Matlab 2016a o superior.
- Driver bebop_autonomy.

Para el hardware se tiene un ordenador de las siguientes características:

- Laptop Toshiba U945 Core i5 tercera generación.
- Disco duro de 1 Tb.
- Memoria Ram 16 Gb.
- Dron Parrot Bebop 1 Version:3.23

Una parte principal del proyecto es la visualización de la información, la cual permitirá corroborar si el sistema de seguimiento de trayectoria cumple con su cometido, es por esto que se desarrolló una Interfaz Humano Maquina o HMI por sus siglas en ingles.

5.2 Interfaz Humano Máquina

La interfaz HMI desarrollada permite el monitoreo del seguimiento de la trayectoria del dron, ya que esta muestra en su pantalla principal la imagen recibida en escala de grises y la imagen en blanco y negro en la cual se puede apreciar la detección de la figura y el centroide de la misma, en la siguiente ventana podemos apreciar los datos del odómetro que provee el dron, con estos datos se visualizará desde una perspectiva superior la trayectoria que fue siguiendo el dron, y por último en la pantalla de ayuda se mostrará la función de cada uno de los botones que permiten la parte de control desde el HMI.

5.2.1 Criterios para el diseño de la interfaz

Para el diseño de la interfaz HMI se consideraron diversos criterios con el fin de que la interfaz fuera práctica al momento de emplearla, sencilla e intuitiva para el operador de tal manera que esta pueda permitir la visualización del seguimiento de la trayectoria.

Los criterios de diseño tomados en cuenta para la realización de la interfaz HMI son (Oscullo, 2018):

- Visibilidad: esta se emplea con el fin de que el operador sea capaz de visualizar las imágenes recibidas por parte del dron.
- Interactividad: para que el usuario sea capaz de desplazar el dron a voluntad o a su vez para dar inicio al programa de seguimiento de trayectoria.
- Proporción de Colores: para poder enfatizar la información más importante dentro de la interfaz y evitar la fatiga visual del operador.

5.2.1.1 Jerarquía de pantallas

Fundamentados en los criterios de diseño, la interfaz fue dividida en 3 pantallas con la finalidad de simplificar la abstracción mental del operador y hacer que la navegación entre pantallas se de en el menor tiempo posible.

En la pantalla principal podemos apreciar la visualización de las imágenes provenientes del dron, así como la imagen binarizada la cual sirve para la detección de la trayectoria, también tenemos las opciones para realizar un control manual o automático del dron. El diseño esquemático de la pantalla principal se le puede apreciar en la Figura 28.

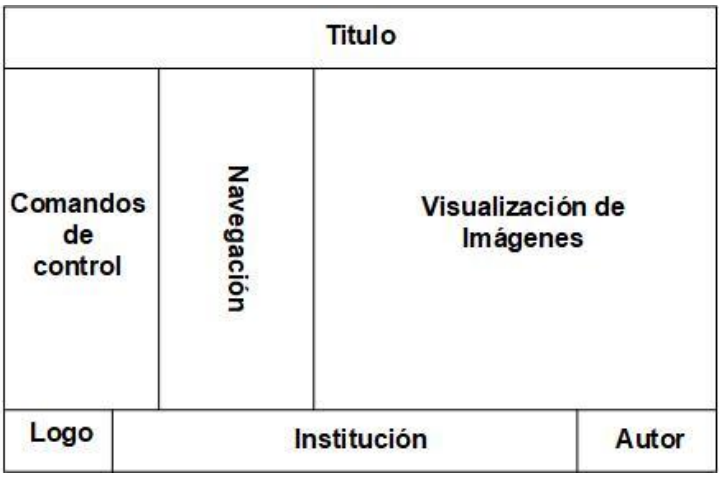


Figura 28. Diseño Esquemático Pantalla Pricipal

La pantalla de odometría, permite observar los valores de odometría que envía el dron durante la ejecución del programa, con la finalidad de apreciar la trayectoria seguida desde una vista superior. El diseño esquemático de la pantalla de odometría se puede apreciar en la Figura 29**.**

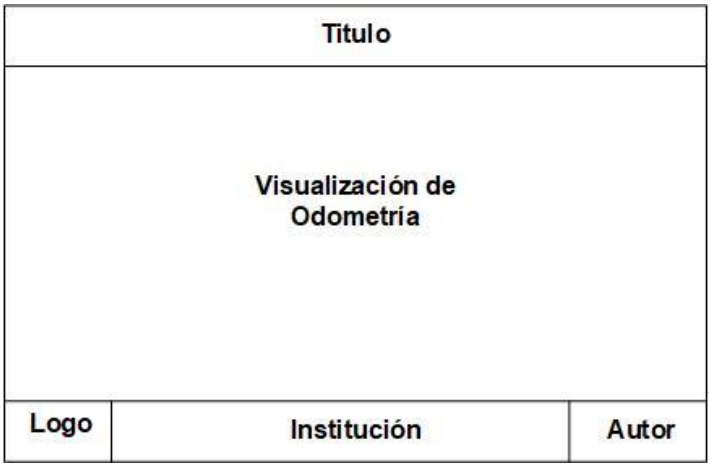


Figura 29. Diseño Esquemático de la Pantalla Odometría

La pantalla de ayuda muestra información necesaria para que el operador pueda manejar la interfaz. El diseño esquemático de la pantalla ayuda se muestra en la Figura 30

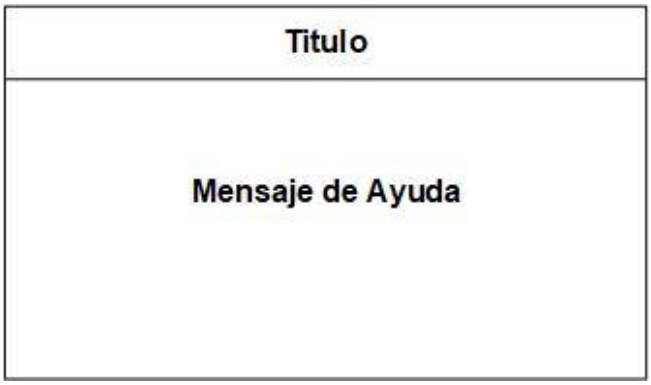


Figura 30. Diseño Esquemático de la Pantalla Ayuda

5.2.1.2 Estructura de las Pantallas

Para evitar la carga visual en el operador o usuario de la interfaz, se recomienda no utilizar combinaciones de colores con contrastes incompatibles, y reforzar la distinción de los colores con elementos como textos o imágenes que tengan relación, para los fondos de pantalla es recomendable utilizar colores neutros como gris, beige, arena, y evitar el uso del blanco o negro debido al resplandor (Yachou, 2014).

Tomando en cuenta los requerimientos anteriores, en la Tabla 14 se pueden apreciar los colores utilizados en las pantallas de la interfaz HMI.

Tabla 14

Colores Uitlizados en la Interfaz

Ítem	Color	Descripción
Fondos de pantalla		Gris claro
Título de pantallas		Azul Marino
Titulo secundario		Amarillo
Texto general		Negro
Botones de navegación		Azul Marino
Botones de control		Gris oscuro
Texto en Botones		Blanco

El desarrollo de la interfaz HMI se lo realizó con la aplicación de todos los parámetros vistos en las secciones anteriores, la aplicación de todos estos parámetros conforman la interfaz HMI del sistema de seguimiento de trayectoria, como se puede apreciar en la Figura 31 tenemos la pantalla principal del HMI, en la Figura 32 tenemos la pantalla correspondiente a la odometría del dron, y en la Figura 33 tenemos la pantalla de ayuda correspondiente al uso de la interfaz.

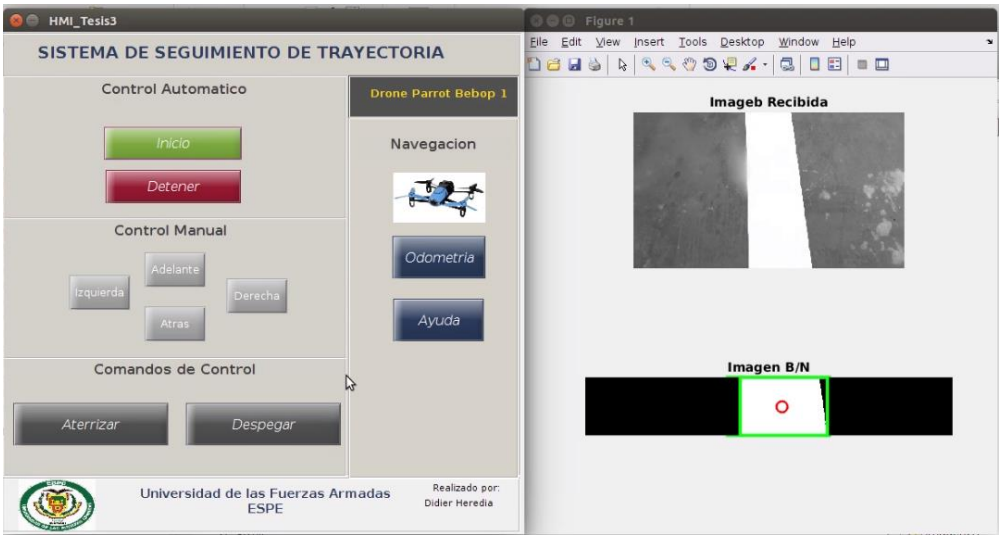


Figura 31. Pantalla principal de la Interfaz

Figura 32. Pantalla de odometría de la Interfaz

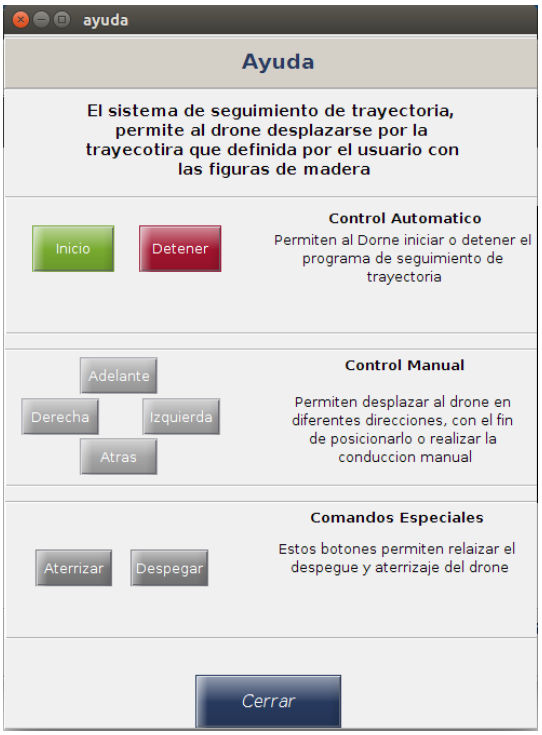


Figura 33. Pantalla de ayuda de la interfaz

CAPÍTULO VI

6. Pruebas y Resultados

6.1 Introducción

Las pruebas de funcionamiento se desarrollaron en dos tipos de ambientes, estos son ambientes interiores dentro de los cuales se tratará de mantener condiciones óptimas como son iluminación adecuada, poca o nula inferencia del viento y en ambientes exteriores dentro del cual tendremos que tomar en cuenta la iluminación natural y el efecto de ráfagas de viento sobre el dron.

6.2 Pruebas en interiores

Las pruebas en interiores se realizaron dentro de una residencia, en un patio delantero que tiene por medidas 5 metros de largo y 3 metros de ancho, este espacio físico resultó suficiente para realizar las pruebas, también para estas pruebas se utilizaron las protecciones PPE exclusivas del dron parrot bebop 1, estas protecciones tienen como objetivo la protección del dron y sus hélices en vuelos realizados en interiores o para principiantes, en la Figura 34 se pueden apreciar las protecciones PPE.

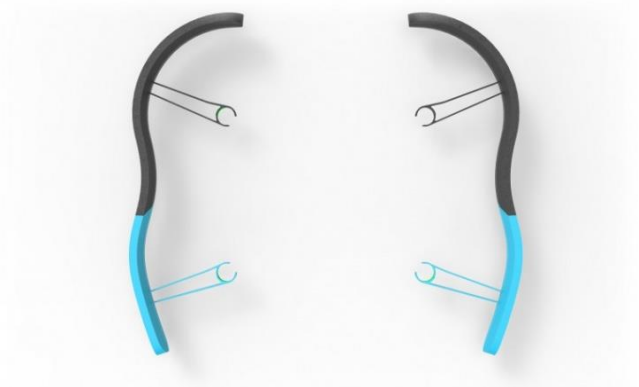


Figura 34. Protecciones PPE parrot bebop 1
Fuente: (Parrot, 2018)

Para las pruebas en interiores se escogieron 2 tipos de trayectorias, las cuales fueron una rectilínea y una trayectoria que presentó giros, esto se lo realizó con el fin de apreciar el

funcionamiento de los controladores difusos ante cada trayectoria. Para la prueba se realizaron un total de 15 vuelos, en las cuales se tomaron en cuenta cuales fueron completadas y cuáles no. Con el fin de conocer cuál es la eficiencia del algoritmo recurrimos a utilizar la Ecuación (1), en esta se toman en cuenta los vuelos realizados y los vuelos completados para realizar el cálculo de la eficiencia.

$$eficiencia = \frac{vuelos\ completados}{vuelos\ realizados} * 100 \qquad \textbf{(1)}$$

Las pruebas en interiores comprendieron un total de 8 trayectoritas rectilíneas y 7 trayectorias con giro, como se puede apreciar en la Tabla 15 tenemos el resultado de los vuelos que se realizaron, la distancia total recorrida y el tiempo que empleó el dron en cumplir cada trayectoria.

Tabla 15

Pruebas realizadas en interiores

N	Trayectoria	Estado	Distancia (m)	Tiempo (min)
1	recta	completada	5	36s
2	recta	completada	5	30s
3	recta	completada	5	22 s
4	recta	completada	5	41s
5	recta	completada	5	25s
6	recta	completada	5	27s
7	recta	completada	5	34s
8	recta	completada	5	36s
9	giro	completada	5	41s
10	giro	completada	5	36s
11	giro	incompleta	5	32s
12	giro	completada	5	33s
13	giro	incompleta	5	38s
14	giro	completada	5	36s
15	giro	completada	5	40s

Con los datos obtenidos de la pruebas en interiores, podemos calcular la eficiencia que el algoritmo, de los datos obtenidos tenemos que de los 15 vuelos desarrollados 13 de estos

cumplieron con la trayectoria, empleando la Ecuación (1) tenemos como resultado una eficiencia del 86.66% para vuelos desarrollados en interiores.

En la Figura 35 se puede apreciar la trayectoria recta realizada en interiores, la cual fue completada en un tiempo de 22 segundos, con una distancia total recorrida de 5 metros lineales.

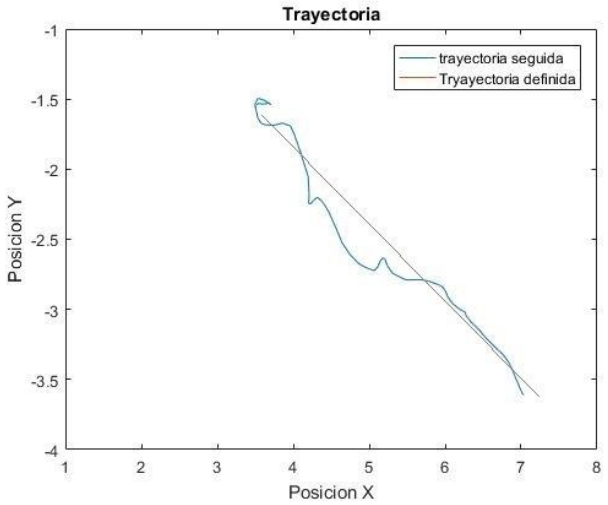


Figura 35. Trayectoria recta en interiores

En la Figura 36 se muestran los resultados obtenidos al momento de realizar el seguimiento a la referencia para las variables de control roll y yaw en la realización de la trayectoria recta realizada en interiores.

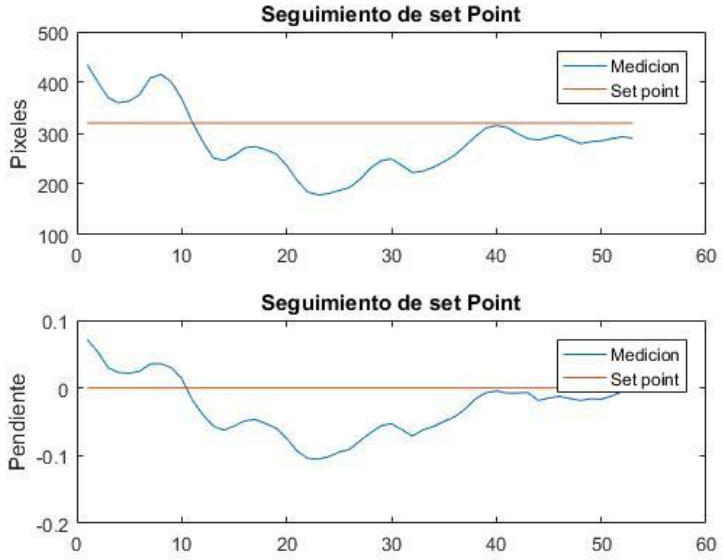


Figura 36. Seguimiento de la referencia trayectoria recta en interiores

Para la trayectoria con giro realizadas en interiores, se optó por hacer que la trayectoria contenga solo un giro, debido al espacio físico limitado que se tiene en el lugar de pruebas, en la Figura 37 se observa la trayectoria con giro realizada por el dron y la trayectoria definida.

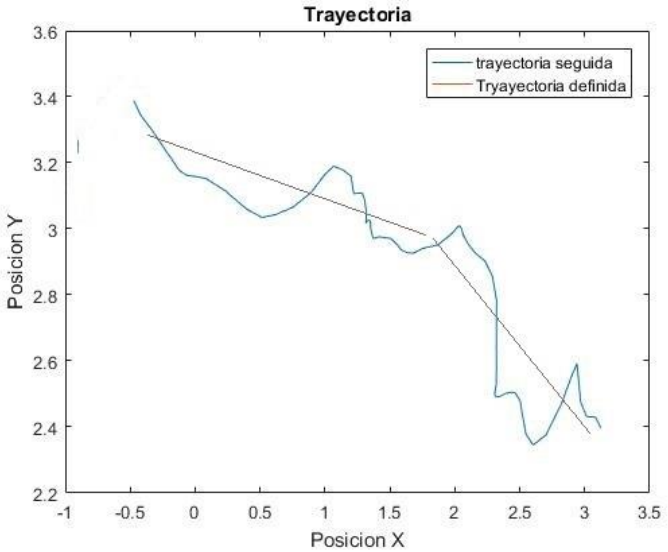


Figura 37. Trayectoria con giro en interiores

En la Figura 38 observamos el seguimiento a la referencia de las variables de control roll y yaw correspondientes a la realización de la trayectoria con giro.

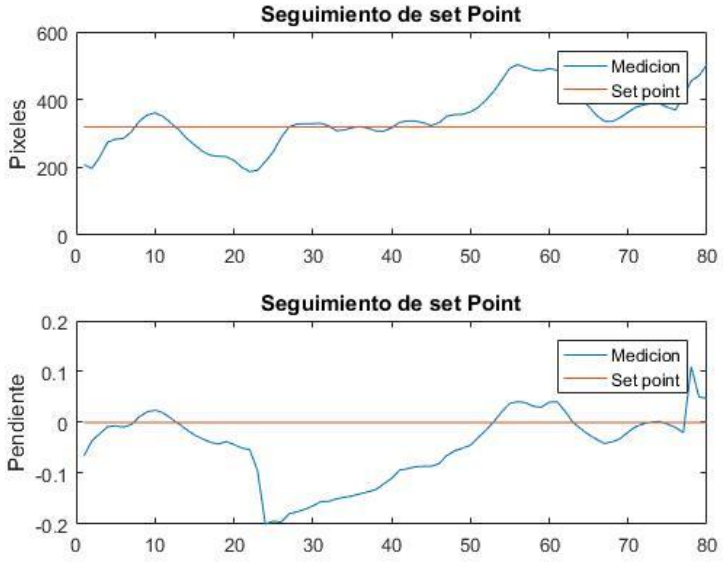


Figura 38. Seguimiento de la referencia trayectoria con giro en interiores

6.3 Pruebas en exteriores

Las pruebas en exteriores se las realizaron en la Universidad de las Fuerzas Armadas ESPE, ubicada en Sangolquí, en el sector de los laboratorios de electrónica, a las afueras del laboratorio de manufactura integrada por computador o CIM por sus siglas en inglés, para estas pruebas se prescindió de las protecciones PPE debido al espacio libre que se tiene, en el cual el dron no podrá chocar con algún objeto cercano.

Al igual que las pruebas para interiores, se realizaron 15 vuelos dentro de las cuales 8 comprenden una trayectoria rectilínea y 7 comprenden una trayectoria con 1 o 2 giros, en estas pruebas se contó con la presencia de viento al momento de realizar la pruebas, puesto que si el viento es demasiado fuerte altera el funcionamiento del dron.

En la Tabla 16 se puede apreciar el resultado de cada una de los vuelos realizadas por el dron, también nos muestra la información referente al estado de la trayectoria y el tiempo que se demoró en cumplir cada una de las trayectorias.

Tabla 16
Prueba realizadas en Exteriores

N	Trayectoria	Estado	Distancia(m)	Tiempo (mim)
1	recta	completada	6	1m 13 s
2	recta	completada	6	1m 1 s
3	recta	completada	6	1m 8s
4	recta	completada	6	1m 12s
5	recta	completada	6	1m 15 s
6	recta	completada	6	1m 8s
7	recta	completada	6	1m 20s
8	recta	completada	6	1m 35s
9	giro	completada	8	1m 7s
10	giro	incompleta	8	19s
11	giro	completada	8	1m 33s
12	giro	completada	8	1m 58s
13	giro	incompleta	8	33s
14	giro	incompleta	8	1m 49s
15	giro	incompleta	8	43s

Con los datos de las pruebas realizadas, se puede calcular la eficiencia que tiene nuestro algoritmo al momento de realizar el seguimiento de las trayectorias, utilizando la Ecuación (1) tenemos que el dron completó 11 trayectorias y un total de 15 vuelos realizados, dándonos una eficiencia del 73.33%, con un mayor acierto en las trayectorias rectas.

A continuación, se presentan las gráficas más representativas de cada una de las trayectorias realizadas por el dron, también se muestra la gráfica del seguimiento al set point para la variable roll y para yaw, con el objetivo de ver cuál fue el comportamiento del dron durante la trayectoria.

En la Figura 39 se aprecia la trayectoria recta realizada en exteriores, la cual se realizó con un viento moderado, y una distancia total recorrida de 6 metros lineales, en un tiempo de 1 minuto y 1 segundo.

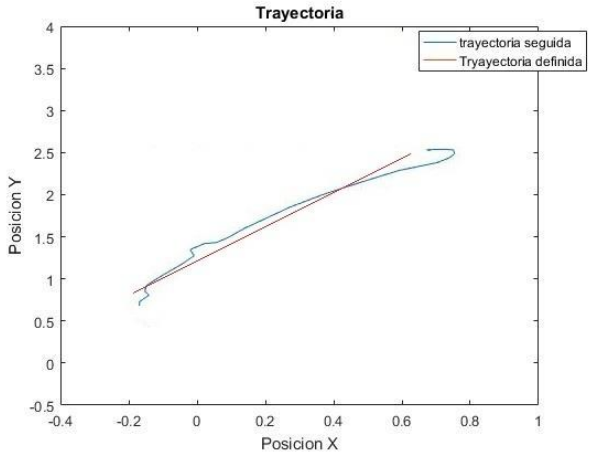


Figura 39. Trayectoria recta en exteriores

En la Figura 40 tenemos el seguimiento a la referencia, esta información corresponde al comportamiento de las variables roll y yaw durante la trayectoria recta realizada en exteriores.

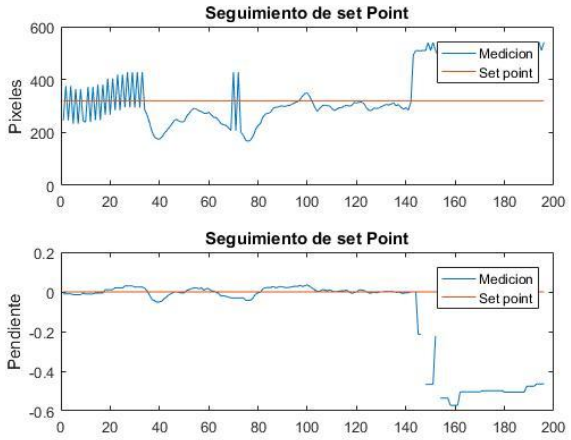


Figura 40. Seguimiento de la referencia trayectoria Recta en exteriores

Para las trayectorias con giro, en exteriores se realizaron trayectorias que contengan 2 giros, en la Figura 41 podemos observar la trayectoria realizada por el dron y compararla con la trayectoria definida, para este caso particular se escogió la trayectoria con el menor tiempo de realización

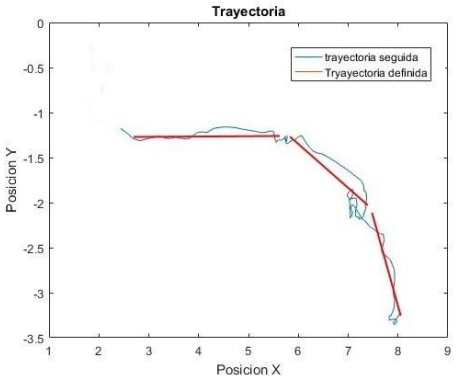


Figura 41. Trayectoria con Giro en exteriores

En la Figura 42 podemos observar el seguimiento a la referencia, con los diversos valores que tomaron las variables de roll y yaw, y apreciar como fue el comportamiento de los controladores para cumplir con la trayectoria.

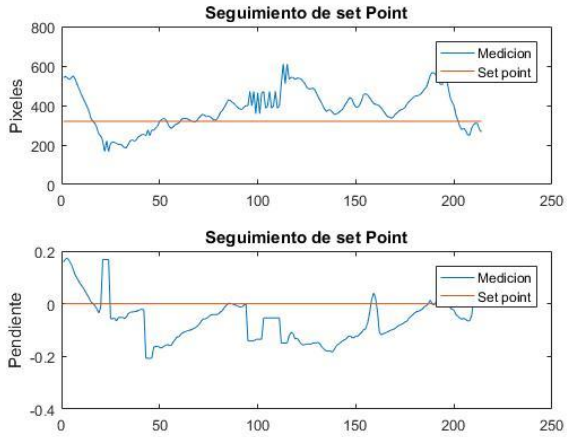


Figura 42. Seguimiento de la referecia trayectoria con giro en exteriores

6.4 Tareas adicionales

Dentro del proyecto se realizaron tareas adicionales correspondientes a la parte del modelado en 3D, debido a inconvenientes presentados con la batería del dron se optó por la fabricación de dos piezas que sirvan de soporte para adaptar baterías li-po de 11.1V y 1300mAh, esto con el fin ampliar el tiempo de vuelo del dron, y tener reemplazos en caso de que las baterías originales

sufran algún desperfecto, en la Figura 43 se puede observar la base de la batería diseñada en SolidWorks y en la Figura 44 podemos observar la base con la betería adaptada.

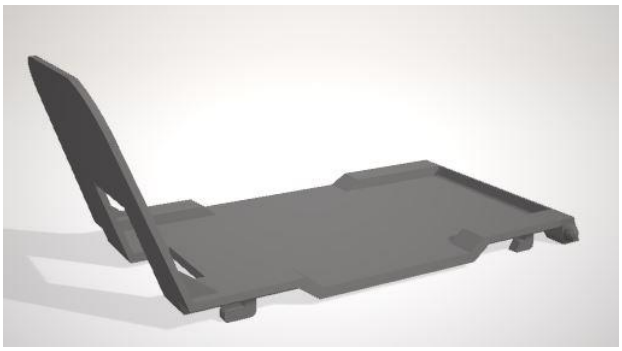


Figura 43. Base para bateria en 3D

Figura 44. Base impresa con bateria Li-po

6.5 Trabajos Futuros

El uso de drones en áreas como la milicia, agricultura, entretenimiento entre otras en el Ecuador es una tecnología en desarrollo (Vargas, 2014), (Terán, 2016). Este trabajo de investigación pretende ampliar el campo de aplicación de los vehículos aéreos no tripulados o drones, respecto al seguimiento autónomo de trayectorias y seguridad, a través del uso de controladores difusos.

Este tipo de controladores permiten obtener una acción de control similar e incluso superior a la de los controladores tradicionales, y su construcción es mucho más fácil (Bora, 2012). Con este proyecto se logró un primer acercamiento en el control de un dron bebop de parrot a través de la

implementación de controladores difusos utilizando Matlab y ROS, ambos comunicándose por medio de un protocolo WiFi. En trabajos futuros se recomienda el uso de un microordenador integrado en el dron con el objetivo de no depender del rango de la red WiFi para establecer la comunicación, logrando de esta manera el seguimiento de trayectorias mucho más largas, Además se propone la implementación de otras técnicas de control como redes neuronales o algoritmos genéticos con el fin de lograr un nivel de autonomía en el dron, para este cometido se pueden recurrir a otros lenguajes de programación como C++ y Python los cuales también trabajan con ROS y librerías de openCV.

CAPÍTULO VII

7. Conclusiones y recomendaciones

7.1 Conclusiones

Se pudo establecer la comunicación entre el dron Bebop de parrot y el ordenador, gracias al meta sistema operativo ROS y al driver bebop autonomy, los cuales fueron instalados en el ordenador de prueba, cabe recalcar que fue muy importante la documentación que se encuentra en la red, ya que para cada versión del sistema operativo Ubuntu existe una versión de ROS dedicada, esto es de mucha ayuda ya que al tratar de instalar una versión de ROS en otra versión de Ubuntu tendremos problemas.

Para la identificación de las variables en primera instancia se trató de utilizar la información de odometría proveniente del dron, pero esta al tener un tiempo de muestreo de muy alto y al obtenerse de una forma dinámica se acordó utilizarlo solo con fines informativos. El otro experimento que se realizó fue utilizar las coordenadas del GPS integrado del dron sin embrago no fue posible utilizarlas debido a que para el dron pueda acceder al GPS es necesario que se conecte a un mínimo de 3 satélites en órbita y este en exteriores, lo cual no iba a permitir realizar las pruebas de funcionamiento en exteriores, y al notar que la resolución del GPS interno era muy baja para distancias cortas se desestimó la idea de utilizar el sensor GPS. Al final se optó por realizar el proyecto con la ayuda de la cámara frontal que posee el dron, las características de la cámara como por ejemplo la estabilización de video y el control de la inclinación mediante software la convirtieron en el sensor ideal para el proyecto.

Para el análisis de las imágenes provenientes del dron primero se utilizó los filtros de Cany y transformadas de Hough con el fin de realizar la detección de los contornos de las piezas de madera y después proceder a la extracción de características, como se puede apreciar en la Figura 45 esos

fueron los resultados obtenidos con estas herramientas. El tiempo de procesamiento provoco un retardo en el análisis de las imágenes del dron por lo que se desestimó el uso de estas herramientas. En base al trabajo realizado por (Lavid, 2017) se optó por utilizar herramientas más simples que permitan realizar el cálculo de la pendiente, área y centroide de las figuras detectadas, mejorando considerablemente la fluidez del video.

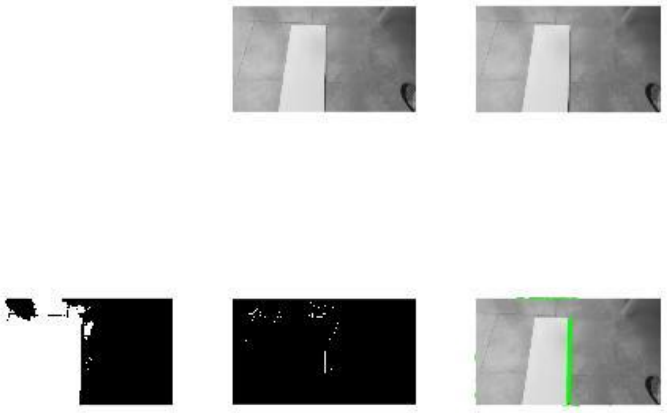


Figura 45. Filtro de Cany y transfromada de Hough

Para el desarrollo e implementación de los controladores difusos se utilizó la herramienta de Matlab *Fuzzy Logic Designer*. Esta herramienta permitió la implementación de un conjunto de reglas que regirán a cada controlador como se pueden apreciar en la Tabla 5 y Tabla 6. Las variables controladas en el dron fueron movimiento lateral o Roll y el movimiento en su propio eje o Yaw. Para la variable del movimiento lateral o en Roll se realizaron pruebas con el controlador Fuzzy PD y Fuzzy PI, obteniendo los mejores movimientos al momento de mantener a la figura detectada en el centro con el Fuzzy PD, lo cual concuerda con el trabajo de (Hongwei, 2015) que sugiere que el controlador fuzzy PD es superior en cuanto al tiempo de respuesta y estabilización. Para la

variable Yaw se realizó la prueba con los dos controladores fuzzy PD y fuzzy PI. En las pruebas de funcionamiento no se obtuvieron los resultados esperados con respecto al giro del dron, debido a que el dron empezó a girar sobre su propio eje de manera errática, con lo cual se optó por implementar un controlador fuzzy proporcional obteniendo mejores resultados al tratar de mantener a la imagen vertical.

Realizando varios vuelos con el dron utilizando el modo manual se pudo determinar las velocidades máximas para cada variable de control, siendo estas 0.3 para roll, 0.2 para yaw y 0.3 para pitch, con estos valores se pudo apreciar que la detección y fluidez del video son óptimas para realizar el seguimiento de la trayectoria.

Con los resultados de las pruebas realizadas en interiores y exteriores, podemos observar que el tiempo de realización de una trayectoria en interiores es menor al tiempo empleado para trayectorias en exteriores, esto se puede deber a la presencia de perturbaciones como pueden ser las ráfagas de viento, y la influencia de la luz natural, la cual en las pruebas en exteriores nos dio algunos falsos positivos en la detección de la figura.

Dentro de la interfaz HMI se pudo notar cierto retraso en el video que enviaba el dron, debido a que para mostrar el video se necesitaba importar las variables desde el workspace base de Matlab, esto generó algunos problemas al momento de la detección y seguimiento de la trayectoria, ya que las acciones de control estaban retrasadas, por esta razón se decidió evitar la importación de las variables al HMI y mostrarlas directamente dentro de una pantalla, con esto se obtuvo una mejor fluidez en el video.

7.2 Recomendaciones

Para la realización de este proyecto se recomienda la utilización de ROS, junto con el driver bebop autonomy, estos se encargan de realizar la comunicación con el dron, y de igual manera se

recomienda el uso de Matlab, el cual permitió realizar cálculos complejos y la implementación de los controladores de una forma más óptima.

Al momento de realizar vuelos en interiores es recomendable el uso de las protecciones PPE del dron, ya que estas ayudaran a proteger la integridad física del dron y también de los usuarios en caso de alguna colisión.

Con el fin de alargar el tiempo de vuelo del dron se imprimieron en 3D bases para adaptar baterías li-po, es recomendable conseguir baterías que tengan el mismo peso que las originales, con el fin de no comprometer la aerodinámica del dron y esto no afecte en el vuelo.

Al momento de realizar vuelos en exteriores es recomendable evitar fuertes ráfagas de viento, debido a que estas pueden de manera considerable el comportamiento del dron, ya que este al ser muy liviano se ve influenciado de forma negativa por los fuertes vientos.

Es recomendable realizar las pruebas en lugares que tengan una muy buena iluminación artificial o natural con el fin de que el proceso del reconocimiento del camino sea el más óptimo y no se den paso a la detección de falsos positivos o a una detección ineficiente.

Bibliografía

Abbasi, E. (2014). Designing Fuzzy PID Controller . *International Journal of Advanced Research in Computer Science & Technol*, 221-227.

Abbasi, E. (2015). Controlling of Quadrotor UAV Using a Fuzzy System for Tuning the PID Gains. *International Journal of Advanced Research in Computer Science & Technol*.

Amazon. (1996-2017). *Amazon Prime*. (Amazon Inc) Recuperado el 27 de 11 de 2017, de Amazon Prime: https://www.amazon.com/Amazon-Prime-Air/b?node=8037720011

Barrientos, A. (2012). Veiculos Aereos No Tripuilados Para su Uso Civil.

Blyenburgh, P. (2006). UAV Systems: Global Review. *Avionics '06 Conference*, 52.

Bora, A. (2012). Design and implementation of a hybrid fuzzy logic controller for a quadrotor VTOL vehicle. *International Journal of Control, Automation and Systems, 10*(1), 61-70.

DJI. (06 de 02 de 2018). *dji Phantom 4*. Obtenido de Dà-Jiāng Innovations: https://www.dji.com/phantom-4-pro?site=brandsite&from=nav

Dong, T. (2005). Path Tracking and Obstacle Avoidance of UAVs - Fuzzy Logic. *Fuzzy Systems, 2005. FUZZ '05. The 14th IEEE International Conference on*.

EPN. (03 de 2017). *Boletin Informativo # 6*. Recuperado el 23 de 11 de 2017, de http://www.epn.edu.ec/wp-content/uploads/2017/04/bol_infor_6.pdf

Fraga, A. (23 de 06 de 2017). *Las 20 companias que dominan el mercado de los drones*. Obtenido de TICbeat: http://www.ticbeat.com/tecnologias/las-20-companias-que-dominan-el-mercado-de-los-drones/

Gutierrez, O. (2017). *Amazon hace su primera entrega por dron en Estados Unidos*. Recuperado el 23 de 11 de 2017, de Cnet: https://www.cnet.com/es/noticias/amazon-prime-air-entrega-dron-estados-unidos/

Hongwei, G. (2015). Fuzzy Adaptive PD Control for Quadrotor Helicopter.

Juste, M. (23 de febrero de 2017). *UPS realiza entrega de paquetes mediante drones*. Recuperado el 23 de 11 de 2017, de Expansion: http://www.expansion.com/economia-digital/companias/2017/02/22/58ad7076e5fdea6b7d8b45ba.html

Kendoul, F. (2012). Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *Journal of Field Robotics, 29*(2), 315–378.

Kore, Y. (1998). *Robotics For Engienners.* New York: McGraw-Hill.

Lavid, C. V. (2017). *Desarrollo de aplicaciones basadas en visión con entorno ROS para el dron Bebop 2.* Madrid.

MathWorks. (2018). *MathWorks Documentation*. Obtenido de https://la.mathworks.com/help/fuzzy/fuzzylogicdesigner-app.html

Mojica, P. (2015). Vehículos aérenos no tripulados, y sus sistemas de comunicación. *Boletin tecnológico CIGEPI* , 104.

Monajjemi, M. (2015). *bebop_autonomy*. Obtenido de http://bebop-autonomy.readthedocs.io/en/latest/index.html

Oscullo, L. (2018). *Modificacion del comportamiento de un Robot simulado en respuesta a los estados de animo en sujetos sanos en base al acople cardiorespiratorio.* Sangolqui.

Parrot. (06 de 02 de 2018). *Parrot bebop 1*. Obtenido de http://global.parrot.com/mx/productos/bebop-drone/

Rodriguez, M. (2016). *Origen y Evolucion de los Drones.* Altran.

ROS. (12 de 03 de 2018). *Robot Operating System*. Obtenido de Robot Operating System: http://www.ros.org

Salcedo, V. (2018). *Aterrizaje automatico de un vehiculo aereo no tripulado basado en el seguimiento de puntos de interes para superficies moviles.* Sangolqui.

Sandoval, F. (17 de Noviembre de 2016). *Los drones sirven de soporte audiovisual*. Obtenido de El Telegrafo: http://www.eltelegrafo.com.ec/noticias/sociedad/4/los-drones-sirven-de-soporte-audiovisual

Sangyam, T. (2010). Path tracking of UAV using self-tuning PID controller based on fuzzy logic. *SICE Annual Conference 2010*.

Santana, E. (03 de 2017). *Tipos de drones y su clasificación*. Obtenido de Xdrones: http://www.xdrones.es/tipos-de-drones-clasificacion-de-drones-categorias-de-drones/

Santos, M. (29 de 07 de 2014). Intelligent Fuzzy Controller of a Quadrotor.

Terán, P. (24 de Octubre de 2016). *Desde el aire, los drones ayudan a mejorar la agricultura local*. Recuperado el 23 de 11 de 2017, de Lideres: http://www.revistalideres.ec/lideres/universidades-drones-agricultura-agroscan.html

Tisdale, J. (2009). Autonomous UAV path planning and estimation. *IEEE Robotics & Automation Magazine, 16*(2).

Vargas, J. (20 de 01 de 2014). *Los drones ya se usan en Ecuador como herramientas de fotografía*. Recuperado el 23 de 11 de 2017, de El Telegrafo: http://www.eltelegrafo.com.ec/noticias/tecnologia/1/los-drones-ya-se-usan-en-ecuador-como-herramientas-de-fotografia

Wackwitz, K. (02 de 10 de 2016). *TOP20 Drone Company Ranking Q3 2016*. Obtenido de Droneii: https://www.droneii.com/top20-drone-company-ranking-q3-2016

Yachou, S. F. (2014). *Aplicación de la guía GEDIS a los Sistemas SCADA del.* La Laguna.

Younes, A. (2008). Attitude stabilization of quadrotor UAV using Backstepping Fuzzy Logic & Backstepping Least-Mean-Square controllers. *Mechatronics and Its Applications, 2008. ISMA 2008. 5th International Symposium on*.